明 細 書

データ処理装置、データ処理プログラム、およびデータ処理プログラムを記録した記録媒体

技術分野

[0001] 本発明は、主記憶手段から命令列および／または値を読み出し、演算処理を行った結果を主記憶手段に書き込む処理を行うデータ処理装置に関するものである。

背景技術

[0002] 従来、CPU(Central Processing Unit)を始めとするマイクロプロセッサにおいて、演算速度の高速化技術に関する研究開発が盛んに行われている。高速化技術としては、例えばパイプライン、スーパースケーラ、アウトオブオーダー実行、および、レジスタリネーミングなどが挙げられる。

[0003] パイプラインは、命令の実行処理を数段階に分解し、複数の命令を流れ作業的に同時処理を行う技術である。スーパースケーラは、命令の実行回路を2組以上用意し、複数の命令を同時に並行して実行する技術である。アウトオブオーダー実行は、命令の記述順序を無視して、いくつかの連続する命令の中から先に実行可能なものを探して先行処理を行う技術である。レジスタリネーミングは、例えばCISC(Complex Instruction Set Computer)タイプのプロセッサにおいて、従来のプロセッサにおける命令の互換性を保ちながら、汎用レジスタの数を増やすことによって並行処理が行われる確率を増大させる技術である。

[0004] このように、マイクロプロセッサにおける演算速度の高速化を図る際には、命令の実行を並行して行うことが重要となっている。しかしながら、プログラム中には、ある命令の結果に応じて異なる命令が行われるような依存関係、言い換えれば分岐が含まれている場合がほとんどである。このような分岐が含まれている場合、並行処理によって先行して処理を行っていると、分岐の結果によって先行処理した内容が無駄になるという状況が発生することになり、演算速度の高速化の効果が小さくなるという問題がある。

[0005] そこで、プログラム中に分岐がある場合に、分岐先を予測することによって先行処

理が無駄になる確率を低減し、並行処理の効果を向上させる技術、いわゆる分岐予測に関する研究が数多く行われている。

[0006] しかしながら、分岐予測に基づいて投機的先行処理を行う場合には、一般的に次のような問題がある。第1の問題としては、予測の正当性を常に検証する必要があるので、先行命令列の実行時間そのものを削減することはできない、という点である。第2の問題としては、誤った予測に基づく一連の先行演算結果を全て無効化する必要があるので、一度に投機的先行処理できる命令数を多くするには、相応のハードウェアコストを要する、という点である。第3の問題としては、命令間の依存関係が多いほど、多重に投機的先行処理をする必要が生じ、予測の正当性の検証処理、および誤った予測に基づく処理の無効化処理が極めて複雑になる、という点である。

[0007] 一方、分岐予測とは異なる高速化技術として、値再利用という技術も提案されている。この値再利用とは、プログラムの一部分に関する入力値および出力値を再利用表に登録しておき、同じ箇所を再度実行する際に、入力値が再利用表に登録されているものである場合には、登録されている出力値を出力する、という技術である。この値再利用による効果としては次のようなものが挙げられる。(1)入力値が、再利用表に登録されている入力値と一致すれば、実行結果を検証する必要がない。(2)入力値および出力値の総数によってのみハードウェアコストが決定され、省略可能な命令列の長さが制約されない。(3)命令間の依存関係の多少は、再利用機構の複雑さに影響を与えない。(4)冗長なロード／ストア命令を削減することができるとともに、これに伴う消費電力の削減も実現される。

[0008] 非特許文献(情報処理学会論文誌:ハイパフォーマンスコンピューティングシステム, HPS5, pp.1-12, Sep.(2002), "関数値再利用および並列事前実行による高速化技術"(中島康彦、緒方勝也、正西申悟、五島正裕、森眞一郎、北村俊明、富田眞治)(発行日2002年9月15日))には、プログラムにおける関数に関して値再利用を行う技術が示されている。この従来技術では、一般的にロードモジュールがABI(Application Binary Interface)に従って作られることを利用しており、特に、SPARC(Scalable Processor ARChitecture) ABIを利用している。そして、このABIにおいて関数の入出力を特定することによって値再利用を実現している。すなわち、値再利用

のためのコンパイラによる専用命令の埋め込みが不要となっており、既存ロードモジュールへの適用が可能となっている。

[0009] また、関数の多重構造を動的に把握することにより、関数内局所レジスタやスタック上の局所変数を値再利用における入出力値から除外するようにしており、これによって効率を向上させている。特に関数については、関数の複雑さに拘わらず、最大6のレジスタ入力、最大4のレジスタ出力、および、局所変数を含まない最小限の主記憶値の登録による再利用および事前実行が可能となっている。この従来技術について以下に詳細に説明する。

[0010] まず、単一の関数を対象として、何が入力で何が出力であるかを明らかにし、1レベルの再利用を行うために必要な機構について説明する。プログラムにおいては、一般的に関数は多重構造を形成している。関数A(Function-A)が関数B(Function-B)を呼び出す構造を図46(a)に示す。

[0011] 大域変数(Globals)は、関数Aの入出力(Ain／Aout)および関数Bの入出力(Bin／Bout)になりうるものである。関数Aの局所変数(Locals-A)は、関数Aの入出力ではないが、ポインタを通じて関数Bの入出力になりうるものである。また、関数Aから関数Bへの引数(Args)は、関数Bへの入力となりうるものであり、関数Bから関数Aの返り値(Ret.Val.)は、関数Bからの出力となりうるものである。なお、関数Bの局所変数(Locals-B)は、関数Aおよび関数Bの入出力には含まれない。

[0012] コンテクストに依存せずに関数Bを再利用するには、関数Bの実行時に、関数Bの入出力Bin／Boutのみを入出力として登録しなければならない。ここで、図46(a)に示すプログラム構造を実行する際の主記憶におけるメモリマップを図46(b)に示す。このメモリマップにおいて、Bin／Boutを含まない領域はLocals-Bのみとなっている。よって、Bin／Boutを識別するには、GlobalsとLocals-Bとの境界、および、Locals-BとLocals-Aとの境界をそれぞれ確定しなければならない。前者については、一般的にOS(Operating System)が実行時のデータサイズおよびスタックサイズの上限を決めることを利用し、OSが設定する境界(LIMIT)に基づいてGlobalsとLocals-Bとの境界を確定することができる。後者については、Bが呼び出される直前のスタックポインタの値(SP in A)を用いることによって、Locals-BとLocals-Aとの境界を確定すること

ができる。

[0013] 次に、与えられた主記憶アドレスが、大域変数であるか、または、どの関数の局所変数であるかを識別する方法について説明する。ロードモジュールは、SPARC ABIに規定されている以下の条件を満たすと仮定する。なお、%fpはフレームポインタ、%spはスタックポインタを意味するものとする。

(1)%sp以上の領域のうち、%sp+0ー63はレジスタ退避領域、%sp+68ー91は引数退避領域であり、いずれも関数の入出力ではない。

(2)構造体を返す場合の暗黙的引数(Implicit Arg.)は%sp+64ー67に格納される。

(3)明示的引数(Explicit Arg.)はレジスタ%o0ー5、%sp+92以上の領域に置かれる。

[0014] まず、大域変数と局所変数とを区別するために、一般的に、OSが実行時のデータサイズおよびスタックサイズの上限を決めることを利用し、次の事項を仮定する。

(1)大域変数はLIMIT未満の領域に置かれる。

(2)%spは、LIMIT以下になることはなく、LIMITー%spの領域は無効である。

[0015] 以上の条件を満たしながら、関数Aが関数Bを呼び出す場合の、メモリマップにおける引数およびフレームの概要を図47に示す。同図を参照しながら、以下にAの局所変数およびBの局所変数を区別する方法について説明する。

[0016] 同図において、(a)はA実行中の状態を示している。LIMIT未満の太枠部分に命令(Instructions)および大域変数(Global Vars.)が格納され、%sp以上に有効な値が格納されている。%sp+64には、Bが構造体を返り値とする場合の暗黙的引数として、構造体の先頭アドレスが格納される。Bに対する明示的引数の先頭6ワードはレジスタ%o0ー5、第7ワード以降は%sp+92以上に格納される。ベースレジスタを%spとするオペランド%sp+92が出現した場合、この領域は引数の第7ワードすなわちBの局所変数である。一方、オペランド%sp+92が出現しない場合、この領域はAの局所変数である。このように、(a)の状態では、オペランドを検証することによってAの局所変数とBの局所変数とを区別することができる。

[0017] 一方、(b)はB実行中の状態を示している。引数が入力、返り値が出力、大域変数

およびAの局所変数が入出力となりうる。ただし、Bは可変長引数を受け入れる場合があるので、一般に%fp＋92以上の領域がAの局所変数の領域となるかBの局所変数の領域となるかは判断できない。

[0018] 局所変数を区別するには、まず、(a)の時点において引数の第7ワード以降を検出した関数呼び出しは再利用の対象外とし、第7ワード以降を検出しない関数呼び出しに関して、直前に%sp＋92の値を記録しておくようにする。なお、第7ワード以降を使用する関数呼び出しの出現頻度が低いと予想されることから、第7ワード以降を使用する関数を再利用の対象外とする制限による性能低下は軽微なものと考える。

[0019] 以上の準備により、(b)における主記憶参照アドレスが、予め記録した%sp＋92の値以上の場合はAの局所変数、小さい場合はBの局所変数であることがわかる。B実行時には、Bの局所変数を除外しながら、大域変数およびAの局所変数を再利用表へ登録する。

[0020] 再利用の際は、Bの局所変数は入出力から除外されるので、Bの局所変数のアドレスが一致している必要がない。このため、いかなるコンテクストであっても、入力さえ一致すれば、再利用することが可能である。ただし、Bが参照する大域変数やAの局所変数については、アドレスおよびデータの両方が再利用表の内容と完全に一致する必要がある。すなわち、Bを実行する前に、どのようにして比較すべき主記憶アドレスを網羅するかがポイントになる。

[0021] Bが参照する大域変数やAの局所変数のアドレスは、そもそもBにおいて生成されるアドレス定数や、大域変数／引数を起源とするポインタに基づいているものである。よって、まず引数が完全に一致する再利用表中のエントリを選択した後に、関連する主記憶アドレスをすべて参照して一致比較を行うことにより、Bが参照すべき主記憶アドレスを網羅することができる。そして、全ての入力が一致した場合にのみ、登録済の出力(返り値、大域変数、およびAの局所変数)を再利用することができる。

[0022] 関数再利用を実現するために、再利用表として、関数管理表(RF)および入出力記録表(RB)を設けることにする。1つの関数を再利用するために必要なハードウェア構成を図48に示す。複数の関数を再利用可能とするには、この構成を複数組用意することになる。

[0023] この表において、RFおよびRBに保持されるVは、エントリが有効であるか否かを示すフラグであり、LRU(least recently used)は、エントリ入れ替えのヒントを示している。RFは、上記のVおよびLRUの他に、関数の先頭アドレス(Start)、および参照すべき主記憶アドレス(Read/Write)を保持する。RBは、上記のVおよびLRUの他に、関数呼び出し直前の%sp(SP)、引数(Args.)(V:有効エントリ、Val.:値)、主記憶値(Mask:Read/Writeアドレスの有効バイト、Value:値)、および、返り値(Return Values)(V:有効エントリ、Val.:値)を保持する。

[0024] 返り値は、%i0ー1(リーフ関数では%o0ー1に読み替える)または%f0ー1に格納され、%f2ー3を使用する返り値(拡張倍精度浮動小数点数)は対象プログラムには存在しないものと仮定する。Readアドレスは RFが一括管理し、MaskおよびValueはRBが管理することにより、Readアドレスの内容とRBの複数エントリをCAM(content-addressable memory)により一度に比較する構成を可能としている。

[0025] 単一の関数を再利用するには、まず、関数実行時に、局所変数を除外しながら、引数、返り値、大域変数および上位関数の局所変数に関する入出力情報を再利用表に登録していく。ここで、読み出しが先行した引数レジスタは関数の入出力として、また、返り値レジスタへの書き込みは関数の出力として登録する。その他のレジスタ参照は登録する必要がない。主記憶参照も同様に、読み出しが先行したアドレスについては入力、書き込みは出力として登録する。

[0026] 関数から復帰するまでに次の関数を呼び出した場合、または、登録すべき入出力が再利用表の容量を超える、引数の第7ワードを検出する、および、途中でシステムコールや割り込みが発生する、などの擾乱が発生しなかった場合、復帰命令を実行した時点で、登録中の入出力表エントリを有効にする。

[0027] 以降、図48を参照しながら説明すると、関数を呼び出す前に、(1)関数先頭アドレスを検索し、(2)引数が完全に一致するエントリを選択し、(3)関連する主記憶アドレスすなわち少なくとも1つのMaskが有効であるReadアドレスをすべて参照して、(4)一致比較を行う。全ての入力が一致した場合に、(5)登録済の出力(返り値、大域変数、およびAの局所変数)を書き戻すことによって、関数の実行を省略することができる。

[0028] ここで、命令区間の一例として、図49に示す命令区間が、図48に示したRFおよびRBの構成によって実行された場合の例について説明する。同図において、PCは、該命令区間が開始された際のPC値を示している。すなわち、命令区間の先頭が1000番地となっている。また、図50は、図49に示す命令区間が実行された場合に、RBに登録される入力アドレスおよび入力データ、並びに出力アドレスおよび出力データを簡略化して示しており、図51は、RBにおける実際の登録状況を示している。

[0029] 第1行目の命令(以降、単に第1の命令のように称する)において、アドレス定数A1がレジスタR0にセットされる。第2の命令において、レジスタR0の内容をアドレスとする主記憶からロードされた4バイトデータ(00110000)がレジスタR1に格納される。この場合、アドレスA1、マスク(FFFFFFFF)(マスクにおいて、Fが有効バイトを示しており、0が無効バイトを示す)、データ(00110000)は、入力としてRBにおけるInput側の第1列に登録され、レジスタ番号R1、マスク(FFFFFFFF)、およびデータ(00110000)は出力としてRBにおけるOutput側の第1列に登録される。

[0030] 第3の命令において、アドレス定数A2がレジスタR0にセットされる。第4の命令において、レジスタR0の内容をアドレスとする主記憶からロードされた1バイトデータ(02)がレジスタR2に格納される。この場合、アドレスA2、マスク(FF000000)、およびデータ(02)は入力としてRBにおけるInput側の第2列に登録される。この際、アドレスA2の残り3バイトについては、Don't Careを意味する「−」が格納される。レジスタ番号R2、マスク(FFFFFFFF)およびデータ(00000002)は出力としてRBにおけるOutput側の第2列に登録される。

[0031] 第5の命令において、アドレス(A2+R2)からロードされた1バイトデータ(22)がレジスタR2に格納されている。アドレスR2の値は(02)であったので、アドレス(A2+02)、およびデータ(22)が、入力としてRBにおけるInput側の第2列に追加登録される。この際、アドレス(A2+02)の部分に登録が行われ、アドレス(A2+01)および(A2+03)に対応する部分は、Don't Careを意味する「−」のままとなる。すなわち、アドレスA2に対応するマスクは(FF00FF00)となる。レジスタ番号R2、マスク(FFFFFFFF)、およびデータ(00000022)は、出力としてRBにおけるOutput側の第2列に上書きされる。

[0032] 第6の命令において、アドレス定数A3がレジスタR0にセットされる。第7の命令において、レジスタR0の内容をアドレスとする主記憶からロードされた1バイトデータ(33)がレジスタR3に格納される。この場合、アドレスA3、マスク(00FF0000)、およびデータ(33)は入力としてRBにおけるInput側の第3列に登録される。レジスタ番号R3、マスク(FFFFFFFF)、およびデータ(00000033)は出力としてRBにおけるOutput側の第3列に登録される。

[0033] 第8の命令において、アドレス(R1+R2)からロードされた1バイトデータ(44)がレジスタR4に格納される。この場合、アドレスR1とアドレスR2は命令区間の内部にて上書きされたレジスタのアドレスとなるので、アドレスR1およびアドレスR2は命令区間の入力とはならない。一方、アドレス(R1+R2)によって生成されたアドレスA4は命令区間の入力であるので、アドレスA4、マスク(00FF0000)、およびデータ(44)は入力としてRBにおけるInput側の第4列に登録される。レジスタ番号R4、マスク(FFFFFFFF)、およびデータ(00000044)は出力としてRBにおけるOutput側の第4列に登録される。

[0034] 第9の命令において、レジスタR5から値が読み出され、読み出された値に1が加えられた結果が再びレジスタR5に格納される。この場合、レジスタR5、マスク(FFFFFFFF)、およびデータ(00000100)は入力としてRBにおけるInput側の第5列に登録される。また、レジスタ番号R5、マスク(FFFFFFFF)、およびデータ(00000101)は出力としてRBにおけるOutput側の第5列に登録される。

[0035] 以上のように、命令実行時におけるメモリ/レジスタからの読み出しに際しては、以下の処理が行われる。

(1)RBにおけるOutput側が検索され、読み出されたアドレス/レジスタ番号が既登録であれば、該アドレス/レジスタ番号はInput側に登録されずに終了する。

(2)RBにおけるOutput側になければRBにおけるInput側が検索され、読み出されたアドレス/レジスタ番号が既登録であれば該アドレス/レジスタ番号は登録されずに終了する。

(3)RBにおけるInput側にもなければ、RBに新たにエントリが追加されて、該アドレス/レジスタ番号および値が登録される。

[0036] また、命令実行時におけるメモリ／レジスタへの書き込みに際しては以下の処理が行われる。

(1)RBにおけるOutput側が検索され、読み出されたアドレス／レジスタ番号が既登録であれば値が更新されて終了する。

(2)RBにおけるOutput側になければ、新たにエントリが追加されて読み出されたアドレス／レジスタ番号および値が登録される。

[0037] また、特許文献(特開2004−258905号公報(公開日2004年9月16日))では、上記のような再利用を行う構成において、プロセッサを複数設け、並列事前実行を行う構成が開示されている。この並列事前実行が行われる際の入力の予測方法として、最後に出現した引数および最近出現した2組の引数の差分に基づいて、ストライド予測を行う方法が開示されている。

[0038] 以上のように入力予測を行えば、上記した入力パラメータが単調に変化し続けるような場合に、事前に予測しておいた結果に基づいて効果的に再利用を行うことが可能となる。

[0039] しかしながら、上記の従来技術では、RBにおいて、各エントリは、1つの項目でも内容が異なれば、それぞれ別のエントリとして登録する必要がある。よって、RBにおけるメモリの利用効率は良くないことになる。また、実行しようとしている関数の入力パターンと、RBの各エントリに含まれている入力パターンとで、1つでも異なるものがあると、再利用を行うことができないことになる。

[0040] また、図52は、図49に示す命令区間が繰り返し実行された場合における、RBの入力側に登録される履歴の例を示している。この例では、Timeが1ー4まで変化するごとに命令区間が実行され、命令区間が実行される度に、アドレスA2の値は、(02)、(03)、(04)、(05)と変化しており、これに伴って他の入力要素における値が変化している。

[0041] また、各履歴の間に示されるdiffは、対応する入力要素の値の変化量を示している。上記した従来の入力予測は、このdiffを用いて予測を行うことになる。図53は、この従来の入力予測による予測結果を示している。

[0042] 例えばループ制御変数のように、単調変化するアドレス(上記の例ではアドレスA2

に対応)の内容については正確に予測することができている。しかしながら、命令区間に配列要素が含まれている場合、配列要素の添字が単調変化していても、配列要素値は一般に単調変化するとは限らない。図52に示す例では、アドレスA2からロードした値が配列要素の添字に該当しており、この添字をアドレスとして用いる主記憶参照はアドレスが変化するために、履歴として登録される入力要素の数そのものが変化することになる。このような状況では、同一列の変化に規則性がなくなるために、図53におけるアドレスA3に対応する列に示すように、予測的中率が極めて悪化することになる。

[0043] 入力予測を行う際に、内容が変化しないアドレスに関する値の予測をすることはハードウェア資源の無駄となる。また、値の変化に規則性がない場合は、差分を0と仮定して予測するしかないが、無理に予測することにより、かえって的中率を下げることがある。図53に示す例では、A2+4に対応するアドレスについてはマスク位置そのものの変化を予測すべきであるが、マスク位置の変化まで予測することは困難である。この場合には、予測せずに直接主記憶値を参照することが得策であることがわかる。

[0044] 以上の課題はいずれも、登録された全てのアドレスを一律に扱ったことにより生じた問題である。

[0045] 本発明は上記の問題点を解決するためになされたもので、その目的は、再利用を行う上でより的確な入出力グループを命令区間記憶手段に登録することを可能とするデータ処理装置を提供することにある。

[0046] 本発明は上記の問題点を解決するためになされたもので、その目的は、比較的簡素な構成によって、再利用を行う上でより的確な入出力グループを命令区間記憶手段に登録することを可能とするデータ処理装置を提供することにある。

発明の開示

[0047] 本発明は上記の問題点を解決するためになされたもので、その第1の目的は、再利用を行う上でより的確な入出力グループを命令区間記憶手段に登録することを可能とするデータ処理装置を提供することにある。

[0048] また、本発明の第2の目的は、主記憶手段から命令列および/または値を読み出し、演算処理を行った結果を主記憶手段に書き込む処理を行うデータ処理装置にお

いて、予測の的中率を向上させることによって、より効果的な命令区間の事前実行を実現するデータ処理装置を提供することにある。

[0049] 本発明に係るデータ処理装置は、上記課題を解決するために、主記憶手段から命令区間を読み出し、演算処理を行った結果を主記憶手段に書き込む処理を行うデータ処理装置において、上記主記憶手段から読み出した命令区間に基づく演算を行う第1の演算手段と、上記第1の演算手段による上記主記憶手段に対する読み出しおよび書き込み時に用いられるレジスタと、上記第1の演算手段によって命令区間の演算が行われたときの入力パターンおよび出力パターンからなる入出力グループを生成する入出力生成手段と、上記入出力生成手段によって生成された入出力グループを記憶する命令区間記憶手段とを備え、上記第1の演算手段が、命令区間を実行する際に、該命令区間の入力パターンと、上記命令区間記憶手段に記憶されている入力パターンとが一致した場合、該入力パターンと対応して上記命令区間記憶手段に記憶されている出力パターンをレジスタおよび／または主記憶手段に出力する再利用処理を行い、上記入出力生成手段が、出力パターンに含まれる各出力要素が、入力パターンに含まれるどの入力要素を起源とするものであるかを示す依存関係格納部と、上記依存関係格納部に格納されている情報に基づいて、1以上の上記出力要素を含む出力パターンと、1以上の上記入力要素を含む入力パターンとからなる入出力グループを設定する入出力グループ設定手段とを備えていることを特徴としている。

[0050] 上記の構成では、第1の演算手段が命令区間を実行する際に、該命令区間の入力パターンと、上記命令区間記憶手段に記憶されている入力パターンとが一致した場合、該入力パターンと対応して上記命令区間記憶手段に記憶されている出力パターンをレジスタおよび／または主記憶手段に出力する再利用処理を行う構成となっている。そして、命令区間記憶手段に記憶される入力パターンおよび出力パターンは、入出力生成手段によって生成されたものとなっている。

[0051] 入出力生成手段は、出力パターンに含まれる各出力要素が、入力パターンに含まれるどの入力要素を起源とするものであるかを示す情報に基づいて、1以上の出力要素を含む出力パターンと、1以上の入力要素を含む入力パターンとからなる入出

力グループを設定し、設定された1以上の入出力グループを生成するようになっている。したがって、ある命令区間が実行された際の入力パターンおよび出力パターンを単純に命令区間記憶手段に登録する場合と比較して、再利用を行う上でより的確な入出力グループを命令区間記憶手段に登録することが可能となる。よって、再利用を行う際の検索効率を向上させることができる。

[0052] また、本発明に係るデータ処理装置は、上記の構成において、上記入出力グループ設定手段が、ある第1の出力要素の起源となる入力要素の組が、他の第2の出力要素の起源となる入力要素の組に全て含まれている場合に、第2の出力要素の起源となる入力要素の組を入力パターン、第1の出力要素および第2の出力要素を出力パターンとする入出力グループを設定する構成としてもよい。

[0053] 上記の構成では、ある第1の出力要素の起源となる入力要素の組が、他の第2の出力要素の起源となる入力要素の組に全て含まれている場合に、これらが1つの入出力グループにまとめられることになる。よって、冗長な入出力グループを削除することが可能となるので、命令区間記憶手段に入出力グループを冗長に登録することを防止することができる。

[0054] また、本発明に係るデータ処理装置は、上記の構成において、上記入出力グループ設定手段が、ある第1の出力要素の起源となる入力要素の組と、他の第2の出力要素の起源となる入力要素の組との間で、共通の入力要素が存在しない場合に、第1の出力要素の起源となる入力要素の組を入力パターン、第1の出力要素を出力パターンとする第1の入出力グループ、および、第2の出力要素の起源となる入力要素の組を入力パターン、第2の出力要素を出力パターンとする第2の入出力グループをそれぞれ設定する構成としてもよい。

[0055] 上記の構成によれば、2つの入出力グループにおいて、共通の入力要素が存在しない場合には、それぞれ別の入出力グループとして設定されることになる。ここで、共通の入力要素が存在しない場合とは、それぞれの入出力グループが互いに依存関係を有さないということになる。すなわち、再利用を行う際に、以前に実行された命令区間における入力パターンおよび出力パターンのうちの一部のみが一致した場合にも、再利用を行うことが可能となるので、再利用が可能となる確率を高めることができ

る。

[0056] また、本発明に係るデータ処理装置は、上記の構成において、上記依存関係格納部が、上記各出力要素を行成分、上記各入力要素を列成分とする2次元配列メモリによって構成され、該2次元配列メモリの各メモリ要素が、該メモリ要素の行成分に対応する出力要素が、該メモリ要素の列成分に対応する入力要素を起源とするか否かの情報を保持している構成としてもよい。

[0057] 上記の構成では、出力パターンに含まれる各出力要素が、入力パターンに含まれるどの入力要素を起源とするものであるかの情報を、2次元配列メモリによって示すようになっている。よって、2次元配列メモリの各メモリ要素に対して、例えば1または0を格納するという単純な処理によって上記の情報を格納することができるとともに、例えば各メモリ要素に関して論理演算を行うことによって、各行成分の関係などを容易に把握することが可能となる。

[0058] また、本発明に係るデータ処理装置は、上記の構成において、上記第1の演算手段によって命令区間の演算が行われる際に、レジスタおよび／または主記憶手段から読み出しが行われた場合に、上記入出力生成手段が、(1)読み出しが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素として依存関係格納部に登録されている場合、該出力要素に対応する依存関係格納部の行成分からなる暫定行列を一時記憶する処理、(2)読み出しが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素としては依存関係格納部に登録されておらず、入力要素として依存関係格納部に登録されている場合、該入力要素に対応する依存関係格納部の列に対応するメモリ要素を1とし、その他のメモリ要素を0とした暫定行列を一時記憶する処理、および、(3)読み出しが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素および入力要素のいずれとしても依存関係格納部に登録されていない場合には、該アドレスおよび値を入力要素として依存関係格納部に登録するとともに、該入力要素に対応する依存関係格納部の列に対応するメモリ要素を1とし、その他のメモリ要素を0とした暫定行列を一時記憶する処理を行い、レジスタおよび／または主記憶手段への書き込みが行われた場合に、上記入出力生成手段が、(4)書き込みが行われたレジスタおよび／または主記憶手段のアドレスが

、出力要素として登録されている場合、登録されている出力要素に対応する出力値を、書き込みが行われた値に更新するとともに、既に登録されている出力要素に対応する依存関係格納部の行成分を、その時点で一時記憶されている全ての暫定行列の論理和に置き換え、その後、一時記憶されている暫定行列を初期化する処理、および、(5)書き込みが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素として登録されていない場合、該アドレスおよび値を出力要素として依存関係格納部に登録するとともに、該出力要素に対応する依存関係格納部の行成分を、その時点で一時記憶されている全ての暫定行列の論理和に置き換え、その後、一時記憶されている暫定行列を初期化する処理を行う構成としてもよい。

[0059] 上記のような処理が行われることによって、ある命令区間が実行された際の入出力関係、すなわち、出力パターンに含まれる各出力要素が、入力パターンに含まれるどの入力要素を起源とするものであるかについての情報を的確に依存関係格納部の2次元配列メモリに格納することができる。

[0060] また、本発明に係るデータ処理装置は、上記の構成において、上記入出力グループ設定手段が、上記2次元配列メモリにおける各行成分間の論理積演算を行う行間論理積比較部を含んでおり、上記入出力グループ設定手段が、依存関係格納部において、ある第1行成分の反転と、ある第2行成分との論理積が全て0になる行成分の組を抽出し、抽出された行成分の組のうち、入力要素の組を最も多く含む行成分以外の行成分を、入出力グループの対象外として設定する構成としてもよい。

[0061] 上記の構成では、各行成分の論理積を行うことによって、入力要素の組を最も多く含む行成分以外の行成分を、入出力グループの対象外として設定するようになっている。この処理によって、ある第1の出力要素の起源となる入力要素の組が、他の第2の出力要素の起源となる入力要素の組に全て含まれている場合に、これらを1つの入出力グループにまとめることが実現されることになる。したがって、冗長な入出力グループを削除することが可能となるので、命令区間記憶手段に入出力グループを冗長に登録することを防止することができる。

[0062] また、本発明に係るデータ処理装置は、上記の構成において、上記入出力グループ設定手段が、上記2次元配列メモリにおける各行成分間の論理積演算を行う行間

論理積比較部を含んでおり、上記入出力グループ設定手段が、依存関係格納部において、他のどの行成分に対しても論理積が全て0になる行成分を、それぞれ入出力グループとして設定する構成としてもよい。

[0063] 上記の構成では、各行成分の論理積を行うことによって、他の行成分に対して独立関係にある行成分を入出力グループとして設定するようになっている。この処理によって、共通の入力要素が存在しない、言い換えれば、互いに依存関係を有さない入出力グループを抽出することができるので、再利用を行う際に、以前に実行された命令区間における入力パターンおよび出力パターンのうちの一部のみが一致した場合にも、再利用を行うことが可能となる。

[0064] また、本発明に係るデータ処理装置は、以上のように、第2の演算手段が、上記第1の演算手段によって処理が行われている命令区間に関して、今後入力が予想される予測入力値に基づいて該命令区間の演算を行い、その結果を上記命令区間記憶手段に対して登録する構成となっていてもよい。この場合、第2の演算手段によって、その時点で第1の演算手段によって処理が行われている命令区間に関して、予測入力値に基づく演算が行われ、その結果が命令区間記憶手段に記憶されることになる。よって、次に、同じ命令区間が出現し、予測入力値と同じ入力が行われた場合には、命令区間記憶手段に記憶されている値を再利用することが可能となる。例えば、入力値が単調に変化するような命令区間の場合には、予測入力値が的中する可能性が高いので、上記の構成による効果は高くなる。

[0065] また、本発明に係るデータ処理装置は、上記の構成において、上記入出力グループ設定手段が、各出力要素が所属する入出力グループの情報を格納する出力側グループ格納部と、各入力要素が所属する入出力グループの情報を格納する入力側グループ格納部と、入出力グループを生成している途中に、上記依存関係格納部に変更があった場合に、変更された出力要素と入力要素との依存関係を格納する一時格納部と、入出力グループを生成している途中に、上記依存関係格納部に変更があった場合に、変更された入出力グループの情報を格納するグループ一時格納部とを備えている構成としてもよい。

[0066] 上記の構成では、第1の演算手段が命令区間を実行する際に、該命令区間の入力

パターンと、上記命令区間記憶手段に記憶されている入力パターンとが一致した場合、該入力パターンと対応して上記命令区間記憶手段に記憶されている出力パターンをレジスタおよび／または主記憶手段に出力する再利用処理を行う構成となっている。そして、命令区間記憶手段に記憶される入力パターンおよび出力パターンは、入出力生成手段によって生成されたものとなっている。

[0067] 入出力生成手段は、出力パターンに含まれる各出力要素が、入力パターンに含まれるどの入力要素を起源とするものであるかを示す情報に基づいて、1以上の出力要素を含む出力パターンと、1以上の入力要素を含む入力パターンとからなる入出力グループを設定し、設定された1以上の入出力グループを生成するようになっている。したがって、ある命令区間が実行された際の入力パターンおよび出力パターンを単純に命令区間記憶手段に登録する場合と比較して、再利用を行う上でより的確な入出力グループを命令区間記憶手段に登録することが可能となる。よって、再利用を行う際の検索効率を向上させることができる。

[0068] ここで、入出力グループ設定手段は、出力側グループ格納部、入力側グループ格納部、一時格納部、およびグループ一時格納部を備えている。すなわち、一時格納部によって、入出力グループの生成処理の途中における、依存関係の履歴を認識することが可能となり、グループ一時格納部によって、入出力グループの生成処理の途中における、入出力グループの履歴を認識することが可能となる。また、これらの情報に基づいて、出力側グループ格納部、および入力側グループ格納部を設定することにより、出力側グループ格納部、および入力側グループ格納部を確認することのみによって、容易に入出力グループの設定処理を行うことが可能となる。よって、複雑な演算処理を行うことなく、比較的小規模な演算手段によって、入出力グループの設定を行うことが可能となる。

[0069] また、本発明に係るデータ処理装置は、上記の構成において、上記入出力グループ設定手段が、入出力グループを生成している途中に、上記出力要素および／または上記入力要素に対して既に割り当てられている入出力グループの情報を格納するグループ管理部をさらに備えている構成としてもよい。

[0070] 上記の構成によれば、グループ管理部によって、入出力グループの生成処理の途

中において、既に使用されている入出力グループを認識することが可能となる。よって、入出力グループの設定処理をより容易に行うことが可能となる。

[0071] また、本発明に係るデータ処理装置は、上記の構成において、上記依存関係格納部が、上記各出力要素を行成分、上記各入力要素を列成分とする2次元配列メモリによって構成され、該2次元配列メモリの各メモリ要素が、該メモリ要素の行成分に対応する出力要素が、該メモリ要素の列成分に対応する入力要素を起源とするか否かの情報を保持している構成としてもよい。

[0072] 上記の構成では、出力パターンに含まれる各出力要素が、入力パターンに含まれるどの入力要素を起源とするものであるかの情報を、2次元配列メモリによって示すようになっている。よって、2次元配列メモリの各メモリ要素に対して、例えば1または0を格納するという単純な処理によって上記の情報を格納することができるとともに、例えば各メモリ要素に関して論理演算を行うことによって、各行成分の関係などを容易に把握することが可能となる。

[0073] また、本発明に係るデータ処理装置は、上記の構成において、上記一時格納部が、上記依存関係格納部における複数行のメモリ要素の論理和を格納するものであり、上記グループ一時格納部が、上記出力側グループ格納部における複数行のメモリ要素の論理和、および／または、上記入力側グループ格納部における複数の入力要素に対応するメモリ要素の論理和を格納するものである構成としてもよい。

[0074] 上記の構成では、一時格納部が、依存関係格納部における複数行のメモリ要素の論理和を格納するものとなっている。よって、入出力グループを生成している途中に、上記依存関係格納部に変更があった場合に、変更された出力要素と入力要素との依存関係を格納するものとしての一時格納部を比較的単純な構成によって実現することができる。また、グループ一時格納部が、出力側グループ格納部および／または入力側番号格納部におけるメモリ要素の論理和を格納するものとなっている。よって、入出力グループを生成している途中に、上記依存関係格納部に変更があった場合に、変更された入出力グループの情報を格納するものとしてのグループ一時格納部を比較的単純な構成によって実現することができる。

[0075] また、本発明に係るデータ処理装置は、上記の構成において、上記入出力グルー

プ設定手段が、入出力グループを生成している途中に、条件分岐命令が検出された場合に、該条件分岐命令が依存する入力要素の情報を格納する条件分岐格納部をさらに備えている構成としてもよい。

[0076] 上記の構成によれば、条件分岐格納部に、条件分岐に関わった入力要素の情報を格納することが可能となる。よって、命令区間の実行時に条件分岐が生じた場合でも、入出力の依存関係を的確に認識することが可能となる。

[0077] また、本発明に係るデータ処理装置は、上記の構成において、上記第1の演算手段によって命令区間の演算が行われる際に、レジスタおよび／または主記憶手段から読み出しが行われた場合に、上記入出力生成手段が、(1)読み出しが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素として依存関係格納部に登録されている場合、該出力要素に対応する依存関係格納部の行成分と、上記一時格納部の各要素との論理和を該一時格納部に格納するとともに、該出力要素に対応する出力側グループ格納部の行成分と、上記グループ一時格納部の各要素との論理和を該グループ一時格納部に格納する処理、(2)読み出しが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素としては依存関係格納部に登録されておらず、入力要素として依存関係格納部に登録されている場合、該入力要素に対応する依存関係格納部の列に対応するメモリ要素を1とし、その他のメモリ要素を0とした情報を上記一時格納部に格納するとともに、該入力要素に対応する入力側グループ格納部の各要素と、上記グループ一時格納部の各要素との論理和を該グループ一時格納部に格納する処理、および、(3)読み出しが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素および入力要素のいずれとしても依存関係格納部に登録されていない場合には、該アドレスおよび値を入力要素として依存関係格納部に登録するとともに、該入力要素に対応する依存関係格納部の列に対応するメモリ要素を1とし、その他のメモリ要素を0とした情報を上記一時格納部に格納する処理を行い、レジスタおよび／または主記憶手段への書き込みが行われた場合に、上記入出力生成手段が、(4)書き込みが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素として登録されている場合、登録されている出力要素に対応する出力値を、書き込みが行われた値に更新するとともに、既に登

録されている出力要素に対応する依存関係格納部の行成分を、その時点で一時記憶されている上記一時格納部に格納されている情報に置き換えるとともに、上記グループ一時格納部に格納されている情報に基づいて、該出力要素に対応する出力側グループ格納部の情報、および、該出力要素が依存する各入力要素に対応する入力側グループ格納部の情報を更新する処理、および、(5)書き込みが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素として登録されていない場合、該アドレスおよび値を出力要素として依存関係格納部に登録するとともに、該出力要素に対応する依存関係格納部の行成分を、その時点で一時記憶されている上記一時格納部に格納されている情報に置き換えるとともに、上記グループ一時格納部に格納されている情報に基づいて、該出力要素に対応する出力側グループ格納部の情報、および、該出力要素が依存する各入力要素に対応する入力側グループ格納部の情報を更新する処理を行う構成としてもよい。

[0078] 上記のような処理が行われることによって、ある命令区間が実行された際の入出力関係、すなわち、出力パターンに含まれる各出力要素が、入力パターンに含まれるどの入力要素を起源とするものであるかについての情報を的確に依存関係格納部の2次元配列メモリに格納することができるとともに、入出力グループの設定を的確に行うことが可能となる。

[0079] また、本発明に係るデータ処理装置は、上記の構成において、上記命令区間記憶手段が、複数の上記入力パターンを、一致比較すべき項目をノードとみなした木構造として記憶する入力パターン記憶手段を備えている構成としてもよい。

[0080] 上記の構成によれば、複数の入力パターンにおいて共通する項目については1つのノードとして記憶することが可能となるので、入力パターン記憶手段における記憶内容の冗長性を低減することが可能となる。したがって、命令区間記憶手段に必要とされる記憶容量を低減することができるので、データ処理装置自体のコストを低減することが可能となる。

[0081] そして、入力パターン記憶手段が、例えば連想検索装置によって構成されている場合、過去の入力パターンがグループ分割されて登録される可能性が高くなっているので、同時に複数の入力パターンの検索が行われる可能性を高めることが可能とな

る。すなわち、一般的な連想検索装置の特性である長レイテンシ高スループットのメリットをより効果的に享受することが可能となる。また、過去の入力パターンがグループ分割されて登録される可能性が高くなることによって、再利用時の入力パターンのヒット率を向上することができる。

[0082] また、本発明に係るデータ処理装置は、上記の構成において、上記入力パターン記憶手段が、上記入力パターンにおいて一致比較すべき項目の値と、次に比較すべき項目とを対応させて格納することによって、上記木構造を実現する構成としてもよい。

[0083] この場合、一致比較すべき項目に関して順に一致比較していくことが可能となるので、一致比較すべき項目をノードとみなした木構造として入力パターンを記憶することを実現することが可能となる。

[0084] また、本発明に係るデータ処理装置は、上記の構成において、上記入力パターン記憶手段が、連想検索手段と、付加記憶手段とを備え、上記連想検索手段が、一致比較すべき項目の値を格納する値格納領域と、該項目を識別するキーを格納するキー格納領域とを有する1つ以上の検索対象ラインを備え、上記付加記憶手段が、上記検索対象ラインに対応した対応ラインごとに、次に連想検索を行うべき項目を格納する検索項目指定領域を有している構成としてもよい。

[0085] この場合、一致比較すべき項目の値が連想検索手段に入力されると、値とキーとが一致する検索対象ラインがシングルマッチし、シングルマッチした検索対象ラインに対応する付加記憶手段における対応ラインによって、次に連想検索を行うべき項目が確定するようになる。

[0086] ここで、各入力パターンは、一致比較すべき項目をノードとみなした木構造として記憶しているので、連想検索手段において、ある項目に関して一致する検索対象は、上記のように1つとなる(シングルマッチ)。シングルマッチ機構のみを有する連想検索メモリは一般的に市販されている一方、マルチマッチを、シングルマッチと同一性能によって報告可能な連想検索メモリは一般的には市販されていない。すなわち、上記の構成によれば、市販の連想検索メモリを連想検索手段として利用することができるので、より短期間かつ低コストで、本発明に係るデータ処理装置を実現することが

可能となる。

[0087] また、上記の課題を解決するために、本発明に係るデータ処理装置は、主記憶手段から命令区間を読み出し、演算処理を行った結果を主記憶手段に書き込む処理を行うデータ処理装置において、上記主記憶手段から読み出した命令区間に基づく演算を行う第1の演算手段と、上記第1の演算手段による上記主記憶手段に対する読み出しおよび書き込み時に用いられるレジスタと、複数の命令区間の実行結果としての入力パターンおよび出力パターンを記憶する入出力記憶手段とを備え、上記第1の演算手段が、命令区間を実行する際に、該命令区間の入力パターンと、上記入出力記憶手段に記憶されている入力パターンとが一致した場合、該入力パターンと対応して上記入出力記憶手段に記憶されている出力パターンをレジスタおよび／または主記憶手段に出力する再利用処理を行うとともに、上記第1の演算手段による命令区間の実行結果を、上記入出力記憶手段に記憶する際に、入力パターンに含まれる入力要素のうち、予測を行うべき入力要素と予測を行う必要のない入力要素とを区別し、この区別情報を上記入出力記憶手段に登録する登録処理手段と、上記区別情報に基づいて、上記入出力記憶手段に記憶されている入力要素のうち、予測を行うべき入力要素の値の変化の予測を行う予測処理手段と、上記予測処理手段によって予測された入力要素に基づいて、該当する命令区間を事前実行する第2の演算手段とをさらに備え、上記第2の演算手段による命令区間の事前実行結果が上記入出力記憶手段に記憶されることを特徴としている。

[0088] 上記の構成では、入出力記憶手段に、複数の命令区間の実行結果としての入力パターンおよび出力パターンが記憶されており、命令区間の実行時に、該命令区間の入力パターンと、入出力記憶手段に記憶されている入力パターンとが一致した場合に再利用を行う構成となっている。そして、予測処理手段によって、入出力記憶手段に記憶されている入力要素の今後の変化が予測され、この予測結果に基づいて、第2の演算手段が命令区間の事前実行を行うようになっている。

[0089] ここで、前記した従来技術のように、単純に入力要素の予測を行うと、予測の的中率が低くなることによって、予測による事前実行の効果が非常に低くなるという問題がある。これに対して、上記の構成によれば、まず登録処理手段によって、入力パター

ンに含まれる入力要素のうち、予測を行うべき入力要素と予測を行う必要のない入力要素とが区別される。そして、予測処理手段は、登録処理手段によって予測を行うべき入力要素と判断された入力要素について予測を行うようになっている。したがって、予測の的中率を向上させることが可能となるので、より効果的な命令区間の事前実行を実現することが可能となる。このような事前実行が行われることによって、次に、同じ命令列が出現し、予測入力値と同じ入力が行われた場合には、命令列記憶手段に記憶されている値を再利用することが可能となる。

[0090] また、本発明に係るデータ処理装置は、上記の構成において、上記登録処理手段が、入力に用いられた上記レジスタの各アドレスに対して、スタックポインタまたはフレームポインタとして用いられる場合、および、該アドレスに対する書き込み命令が定数セット命令である場合に、該当アドレスに対して区別情報として定数フラグをセットし、上記以外の場合に、該当アドレスに対して上記定数フラグをリセットする構成としてもよい。

[0091] 上記の構成によれば、入力に用いられたレジスタのアドレスのうち、アドレスが固定しており、かつ、値が単調変化すると予測されるアドレスに定数フラグをセットすることが可能となる。よって、定数フラグがセットされているレジスタのアドレスに基づく入力要素に対して予測を行うようにすることによって、予測的中率を向上させることが可能となる。

[0092] また、本発明に係るデータ処理装置は、上記の構成において、上記登録処理手段が、入力要素が新規に上記入出力記憶手段に記憶される際に、該入力要素のアドレスに対して、区別情報として変更フラグをリセットし、上記入出力記憶手段に記憶された後に、該当アドレスに対してストア命令が実行された場合に、該当アドレスに対して変更フラグをセットする構成としてもよい。

[0093] 上記の構成によれば、入出力記憶手段に記憶されたものの、その後一度も書き込みが行われないアドレスに対しては、変更フラグがリセットされた状態となる。このようなアドレスに記憶されている内容は変化していないことになるので、該アドレスに対して予測を行う必要はないことになる。すなわち、上記のような変更フラグが入力要素のアドレスに設けられることによって、予測が必要なアドレスのみに対して予測を行う

ことが可能となる。よって、予測処理のためのハードウェア資源を有効に利用することが可能となる。

[0094] また、本発明に係るデータ処理装置は、上記の構成において、上記登録処理手段が、入力要素が新規に上記入出力記憶手段に記憶される際に、該入力要素のアドレスに対して、区別情報として履歴フラグをリセットし、該アドレスに対するロード命令実行時に、該アドレスを生成したレジスタアドレスに上記定数フラグがセットされている場合に、該アドレスに対して履歴フラグをセットする構成としてもよい。

[0095] 上記の構成によれば、入出力記憶手段に記憶されている入力要素のアドレスに対するロード命令実行時に、該アドレスを生成したレジスタアドレスに上記定数フラグがセットされている場合に、該アドレスに対して履歴フラグがセットされるようになっている。ここで、定数フラグがセットされているレジスタアドレスとは、上記のように、アドレスが固定しており、かつ、値が単調変化すると予測されるアドレスとなっている。よって、このようなレジスタアドレスに基づいて生成されたアドレスに関して予測を行うことによる予測的中率は高くなることが予想される。すなわち、上記のような履歴フラグを設けることによって、予測すべきアドレスを適切に設定することが可能となる。

[0096] なお、履歴フラグとしては、各アドレスに文字通りのフラグをたてるようにしてもよいし、複数のバイトデータからなるアドレスのうち、履歴保存対象とするバイト位置を示すマスクといった形式で履歴フラグを実現するようにしてもよい。

[0097] また、本発明に係るデータ処理装置は、上記の構成において、上記登録処理手段が、入力要素が新規に上記入出力記憶手段に記憶される際に、該入力要素のアドレスに対して、区別情報として変更フラグをリセットし、上記入出力記憶手段に記憶された後に、該当アドレスに対してストア命令が実行された場合に、該当アドレスに対して変更フラグをセットするとともに、上記予測処理手段が、上記入出力記憶手段に記憶されている入力要素のアドレスのうち、上記変更フラグがセットされ、かつ、履歴フラグがセットされているアドレスに関して、入力要素の変化の予測を行う構成としてもよい。

[0098] ここで、変更フラグがセットされているアドレスとは、上記したように、予測を行うことによる効果が期待できるアドレスとなる。また、履歴フラグがセットされているアドレスと

は、上記したように、予測的中率が高いことが期待できるアドレスとなる。したがって、上記の構成によれば、予測を行うことによる効果が高いと予想されるアドレスに関してのみ予測が行われることになる。よって、予測処理のためのハードウェア資源を有効に利用することが可能となる。

[0099] また、本発明に係るデータ処理装置は、上記の構成において、上記予測処理手段が、上記入出力記憶手段に記憶されている入力要素のうち、該入力要素の履歴における値の変化量が0ではない入力要素のみに対して、入力要素の値の変化の予測を行う構成としてもよい。

[0100] 上記の構成によれば、履歴における値の変化量が0ではない入力要素のみに対して、入力要素の値の変化の予測が行われることになる。ここで、履歴における値の変化量が0となっている入力要素とは、変化がないことが予想される入力要素であるので、該入力要素に対して予測を行う必要はないことになる。すなわち、上記の構成によれば、予測が必要なアドレスのみに対して予測を行うことが可能となる。よって、予測処理のためのハードウェア資源を有効に利用することが可能となる。

[0101] また、本発明に係るデータ処理装置は、上記の構成において、上記登録処理手段が、上記第1の演算手段による命令区間の実行結果を、上記入出力記憶手段に記憶する際に、入力パターンに含まれる入力要素のうち、予測を行うべき入力要素と予測を行う必要のない入力要素とを区別し、この区別情報を上記入出力記憶手段に登録するとともに、上記入出力記憶手段に格納される出力パターンにおける出力要素のうち、該当命令区間の実行の際にストアが行われたものについてそのストアの回数をカウントし、このカウント値を上記入出力記憶手段に格納し、上記第2の演算手段が、上記予測処理手段によって予測された入力要素に基づいて、該当する命令区間を事前実行するとともに、上記カウント値に基づいて該当入力要素に対して行われるストアの回数を待機した上で主記憶からの読み出しを行って該当する命令区間の事前実行を行う構成としてもよい。

[0102] 上記の構成では、登録処理手段は、入出力記憶手段に格納される出力パターンにおける出力要素のうち、該当命令区間の実行の際にストアが行われたものについてそのストアの回数をカウントし、このカウント値を入出力記憶手段に格納する。そして、

予測処理手段は、上記カウント値に基づいて該当入力要素に対して行われるストアの回数を待機した上で主記憶からの読み出しを行って該当する命令区間の事前実行を行うようになっている。したがって、例えば値の変化が不定となる出力要素に関しては予測を行うことが困難であり、この場合に上記のようにカウントされたストアの回数を待機した上で主記憶読み出しが行われることによって、適切な入力要素の値を設定した状態で事前実行を行うことが可能となる。

[0103] 以上のような構成により、より的確な事前実行を実現することが可能となる。このような事前実行が行われることによって、次に、同じ命令列が出現し、予測入力値と同じ入力が行われた場合には、入出力記憶手段に記憶されている値を再利用することが可能となる可能性を一段と高めることができる。

[0104] また、本発明に係るデータ処理装置は、上記の構成において、上記入出力記憶手段が、上記第1の演算手段による命令区間の実行結果としての入力パターンおよび出力パターンを一時的に記録する入出力記録領域を備え、上記入出力記録領域が、各出力要素に対して、ストアが行われた回数を格納するストアカウンタを有する構成としてもよい。

[0105] 上記の構成によれば、入出力記憶手段に入出力記録領域が設けられており、この入出力記録領域に、各出力要素に対して、ストアが行われた回数を格納するストアカウンタが設けられている。これにより、第1の演算手段によって命令区間の実行が行われた際に、該命令区間の実行に際して、各出力要素に対して行われたストアの回数を的確に記録することが可能となる。

[0106] また、本発明に係るデータ処理装置は、上記の構成において、上記入出力記憶手段が、上記第1の演算手段によって演算が行われた命令区間毎に過去の実行結果の履歴を格納する履歴格納領域を備え、上記登録処理手段が、上記入出力記録領域に記録された実行結果を上記履歴格納領域に格納するとともに、上記入出力記録領域に記録された実行結果の入力パターンに含まれる入力要素のうち、履歴格納領域に前回の実行結果として登録されている出力要素と同じアドレスの入力要素に対して、対応する前回の出力要素のストアカウンタを該入力要素に対するストアカウンタとして登録する構成としてもよい。

[0107] 上記の構成によれば、まず入出力記録領域に記録された実行結果が順次命令区間毎に設けられた履歴格納領域に格納される。そして、入出力記録領域から履歴格納領域に格納される入力パターンに含まれる入力要素のうち、履歴格納領域に前回の実行結果として登録されている出力要素と同じアドレスの入力要素に対して、対応する前回の出力要素のストアカウンタが該入力要素に対するストアカウンタとして登録される。ここで、履歴格納領域に格納される入力要素のうち、前回の実行結果としての出力要素と同じアドレスとなる入力要素は、前回の実行結果に影響を受ける入力要素となる。すなわち、このような入力要素に対して上記のようにストアカウンタを設定することによって、該当入力要素に対して予測を行う際に、待機すべきストアの回数を的確に設定することが可能となる。

[0108] また、本発明に係るデータ処理装置は、上記の構成において、上記入出力記憶手段が、上記予測処理手段によって予測された入力要素を格納する予測値格納領域を備え、上記予測処理手段が、上記履歴格納領域に格納されている入力要素のうち、実行履歴の間での値の変化量が一定である入力要素に関して値の予測を行い、上記予測値格納領域に格納する構成としてもよい。

[0109] 上記の構成によれば、まず入出力記憶手段に予測値格納領域が設けられている。そして、予測処理手段が、実行履歴の間での値の変化量が一定である入力要素に関して値の予測を行い、予測値格納領域に格納する。ここで、履歴における命令区間の実行結果の間での値の変化量(差分)が一定である入力要素は、今後もその変化量が一定である可能性が高いものであるので、これに基づいて予測を行うことが可能である。このようにして予測を行った結果を予測値格納領域に格納することによって、予測的中の可能性の高い予測値を設定することが可能となる。

[0110] また、本発明に係るデータ処理装置は、上記の構成において、上記入出力記憶手段が、ストアの回数を待機した上で主記憶からの読み出しを行うべき入力要素を格納する待機要アドレス格納領域を備え、上記予測処理手段が、上記履歴格納領域に格納されている入力要素のうち、実行履歴においてアドレスが変化せず、実行履歴の間での値の変化量が不定である入力要素に関して、上記ストアカウンタ、および予測距離に基づく値としての待機カウンタを上記待機要アドレス格納領域に格納する

構成としてもよい。

[0111] 上記の構成によれば、まず入出力記憶手段に待機要アドレス格納領域が設けられている。そして、予測処理手段が、実行履歴においてアドレスが変化せず、実行履歴の間での値の変化量が不定である入力要素に関して、上記ストアカウンタ、および予測距離に基づく値としての待機カウンタを待機要アドレス格納領域に格納する。ここで、予測距離とは、該当命令区間が今後繰り返し実行された場合の、現時点からの実行回数を示している。実行履歴においてアドレスが変化せず、実行履歴の間での値の変化量が不定である入力要素とは、該当アドレスに対して、命令区間が繰り返し実行される毎にストアが行われるものである。よって、待機カウンタを、上記のようにストアカウンタおよび予測距離に基づいて設定することによって、待機すべき回数を適切に設定することが可能となる。

[0112] また、本発明に係るデータ処理装置は、上記の構成において、上記入出力記憶手段が、ストアの回数を待機した上で主記憶からの読み出しを行うべき入力要素を格納する待機要アドレス格納領域を備え、上記予測処理手段が、上記履歴格納領域に格納されている入力要素のうち、実行履歴においてアドレス自体が変化し、それぞれのアドレスの値も、ストアが発生することにより変化する入力要素に関して、上記ストアカウンタに基づく値としての待機カウンタを上記待機要アドレス格納領域に格納する構成としてもよい。

[0113] 上記の構成によれば、まず入出力記憶手段に待機要アドレス格納領域が設けられている。そして、予測処理手段が、実行履歴においてアドレス自体が変化し、それぞれのアドレスの値も、ストアが発生することにより変化する入力要素に関して、上記ストアカウンタに基づく値としての待機カウンタを上記待機要アドレス格納領域に格納する。実行履歴においてアドレス自体が変化し、それぞれのアドレスの値も、ストアが発生することにより変化する入力要素とは、命令区間が繰り返し実行される毎にアドレスが変化し、また、値の変化量も不定となるものである。よって、待機カウンタを、上記のようにストアカウンタのみに基づいて設定することによって、待機すべき回数を適切に設定することが可能となる。

[0114] また、本発明に係るデータ処理装置は、上記の構成において、第2の演算手段が

主記憶手段から値を読み出す際に、上記予測値格納領域において、ストアカウンタ値がセットされておらず、予測値が有効である場合に該予測値を読み出し値とし、ストアカウンタが0よりも大きい場合にはストアカウンタが0になるまで待機し、ストアカウンタが0になった時点で値を取り出す構成としてもよい。

[0115] また、本発明に係るデータ処理装置は、上記の構成において、第2の演算手段が主記憶手段へ値を書き込む際に、他の第2の演算手段に対して書き込みアドレスおよび値を通知するとともに、該通知を受信した他の第2の演算手段は、予測値格納領域に同一アドレスが登録されている場合に、該入力要素のストアカウンタを1だけ減じて書き込み値を格納し、ストアカウンタが既に0である場合には何も行わない構成としてもよい。

図面の簡単な説明

[0116] [図1]本発明の一実施形態に係るデータ処理装置が備える命令区間記憶部の概略構成を示す図である。

[図2]上記データ処理装置の概略構成を示すブロック図である。

[図3]上記命令区間記憶部における連想検索動作の具体例を示す図である。

[図4(a)]図4(b)における連想検索動作を木構造として示す図である。

[図4(b)]上記命令区間記憶部における連想検索動作の他の具体例を示す図である。

[図5(a)]図5(b)における連想検索動作を木構造として示す図である。

[図5(b)]上記命令区間記憶部における連想検索動作のさらに他の具体例を示す図である。

[図6]関数およびループが入れ子構造となっている状態の一例を示す図である。

[図7]関数の入れ子構造において、内側の構造のレジスタ入出力が、外側の構造のレジスタ入出力となる影響範囲を示す図である。

[図8]比較例におけるRFおよびRBの概略構成を示す図である。

[図9]比較例における検索動作の例を示す図である。

[図10]第2構成例としてのRWの概略構成を示す図である。

[図11]命令区間の一例を示す図である。

[図12]RWの第1構成例におけるメモリ構成の概略を示す図である。

[図13]RWの第1構成例によって生成された入出力セットが木構造として登録された状態を示す図である。

[図14]RWの第2構成例によって生成された入出力セットが木構造として登録された状態を示す図である。

[図15]命令区間の一例を示す図である。

[図16]第1構成例としてのRWのメモリ構成の概略を示す図である。

[図17]RWの第1構成例によって生成された入出力セットが木構造として登録された状態を示す図である。

[図18]図16に示す入出力セットに対して、互いに独立な入力セットおよび出力セットそれぞれにグループ番号を付与した状態を示す図である。

[図19]グループ番号に基づいて、図17に示す木構造を複数の木構造に分割した状態を示す図である。

[図20]第3構成例としてのRWの概略構成を示す図である。

[図21]第4構成例としてのRWの概略構成を示す図である。

[図22]命令区間のさらに他の一例を示す図である。

[図23]グループ番号に基づいて生成された複数の木構造が格納された上記命令区間記憶部における連想検索動作の具体例を示す図である。

[図24]本発明の他の実施形態に係るデータ処理装置が備えるRF／RBによって実現される再利用表を示す図である。

[図25]上記データ処理装置の概略構成を示すブロック図である。

[図26]命令がデコードされた結果、関数呼び出し命令である場合の処理の流れを示すフローチャートである。

[図27]命令がデコードされた結果、関数復帰命令である場合の処理の流れを示すフローチャートである。

[図28]命令がデコードされた結果、後方分岐成立である場合の処理の流れを示すフローチャートである。

[図29]命令がデコードされた結果、後方分岐不成立である場合の処理の流れを示すフローチャートである。

[図30]RWと、RF・RBとの関係を示す図である。

[図31]ある命令区間が実行された場合のRBにおける実際の登録状況を示す図である。

[図32]ある命令区間が繰り返し実行された場合における、履歴としてRBに登録された例を示す図である。

[図33]予測に基づいて、予測処理部がアドレスA2およびアドレスR5の値に関して予測を行った場合の、予測エントリとしてRBに記録される入力要素の状態を示す図である。

[図34]本発明のさらに他の実施形態に係るデータ処理装置が備える命令区間記憶部におけるRFおよびRBの構成の概要を示す図である。

[図35]上記データ処理装置の概略構成を示すブロック図である。

[図36(a)]命令区間の一例を示す図である。

[図36(b)]図36(a)に示す命令区間が実行された場合に、RBに登録される入力アドレスおよび入力データ、並びに出力アドレスおよび出力データを簡略化して示す図である。

[図36(c)]図36(a)に示す命令区間に引き続いて行われる第2回目のループ処理の例を示す図である。

[図36(d)]図36(c)におけるRBに登録される入力アドレスおよび入力データ、並びに出力アドレスおよび出力データを簡略化して示す図である。

[図36(e)]図36(c)に示す命令区間に引き続いて行われる第3回目のループ処理の例を示す図である。

[図36(f)]図36(e)におけるRBに登録される入力アドレスおよび入力データ、並びに出力アドレスおよび出力データを簡略化して示す図である。

[図37]図36(a)に示す命令区間が実行された場合のRBにおける実際の登録状況を示す図である。

[図38(a)]図36(a)に示す命令区間が繰り返し実行された場合における、履歴としてRBに登録された例を示す図である。

[図38(b)]予測処理部がアドレスA1の値に関して予測を行った場合の、予測エントリと

してRBに記録される入力要素の状態を示す図である。

[図39]参考例による予測に基づいて、ループ処理の2回目および3回目における事前実行を行った結果を示す図である。

[図40(a)]RBにおける入出力記録行の例を示す図である。

[図40(b)]履歴格納行の例を示す図である。

[図41(a)]図36(a)に示す命令区間が繰り返し実行された場合における、履歴格納行の登録例を示す図である。

[図41(b)]図41(a)に示す履歴に基づいて、予測処理部が以下に示す予測処理を行った際の、予測値格納領域および待機要アドレス格納領域の例を示す図である。

[図42]予測値に基づく事前実行を行う場合の実行例を示す図である。

[図43]命令区間記憶部の第2の構成例の概略を示す図である。

[図44]図43に示す命令区間記憶部における連想検索動作の具体例を示す図である。

[図45]第2の構成例を適用した場合のデータ処理装置の概略構成を示す図である。

[図46(a)]関数Aが関数Bを呼び出す構造を概念的に示す概念図である。

[図46(b)]図46(a)に示すプログラム構造を実行する際の主記憶におけるメモリマップを示す図である。

[図47]関数Aが関数Bを呼び出す場合の、メモリマップにおける引数およびフレームの概要を示す図である。

[図48]1つの関数を再利用するための従来の再利用表を示す図である。

[図49]命令区間の一例を示す図である。

[図50]図49に示す命令区間が実行された場合に、RBに登録される入力アドレスおよび入力データ、並びに出力アドレスおよび出力データを簡略化して示す図である。

[図51]RBにおける実際の登録状況を示す図である。

[図52]図49に示す命令区間が繰り返し実行された場合における、RBの入力側に登録される履歴の例を示す図である。

[図53]従来の入力予測による予測結果を示す図である。

発明を実施するための最良の形態

[0117] ＜実施の形態1＞

本発明の一実施形態について図面に基づいて説明すると以下の通りである。

[0118] (データ処理装置の構成)

本実施形態に係るデータ処理装置の概略構成を図2に示す。同図に示すように、該データ処理装置は、MSP(Main Stream Processor)1A、SSP(Shadow Stream Processor)1B、再利用表としての命令区間記憶部(命令区間記憶手段)2、および主記憶(主記憶手段)3を備えた構成となっており、主記憶3に記憶されているプログラムデータなどを読み出して各種演算処理を行い、演算結果を主記憶3に書き込む処理を行うものである。なお、同図に示す構成では、SSP1Bを1つ備えた構成となっているが、2つ以上備えた構成となっていてもよい。また、同図に示す構成では、SSP1Bを備えた構成となっているが、SSP1Bを備えていない構成としてもかまわない。SSP1Bを備えた場合の作用・効果については、後述する。

[0119] 命令区間記憶部2は、プログラムにおける関数やループなどの命令区間を再利用するためのデータを格納するメモリ手段である。この命令区間記憶部2の詳細については後述する。

[0120] 主記憶3は、MSP1AおよびSSP1Bの作業領域としてのメモリであり、例えばRAM(Random Access Memory)などによって構成されるものである。例えばハードディスクなどの外部記憶手段や、外部のI／O(input/output)装置などの外部装置からプログラムやデータなどが主記憶3に読み出され、MSP1AおよびSSP1Bは、主記憶3に読み出されたデータに基づいて演算を行うことになる。また、MSP1Aによる演算結果が主記憶3に書き込まれ、この演算結果が上記外部装置に送出されることになる。

[0121] MSP1Aは、再利用記憶手段としてのRW(入出力生成手段)4A、演算器(第1の演算手段)5A、レジスタ6A、およびCache7Aを備えた構成となってい「る。また、SSP1Bは、同様に、再利用記憶手段としてのRW(第2の演算手段)4B、演算器(第2の演算手段)5B、レジスタ6B、およびCache／Local7Bを備えた構成となっている。

[0122] RW4A・4Bは、再利用ウィンドウであり、現在実行中かつ登録中であるRF(付加記憶手段)およびRB(連想検索手段)(後述する)の各ラインをリング構造のスタックとして保持するものである。このRW4A・4Bは、実際のハードウェア構造としては、命令

区間記憶部2における特定のラインをアクティブにする制御線の集合によって構成される。また、詳細は後述するが、RW4A・4Bは、実行された命令区間に関して入出力パターンを生成し、この生成された入出力グループを命令区間記憶部2に対して実行結果として登録する処理を行う。

[0123] 演算器5A・5Bは、レジスタ6A・6Bに保持されているデータに基づいて演算処理を行うものであり、ALU(arithmetic and logical unit)と呼ばれるものである。レジスタ6A・6Bは、演算器5A・5Bによって演算を行うためのデータを保持する記憶手段である。なお、本実施形態では、演算器5A・5B、およびレジスタ6A・6Bは、SPARCアーキテクチャに準じたものとする。Cache7A・7Bは、主記憶3と、MSP1AおよびSSP1Bとの間でのキャッシュメモリとして機能するものである。なお、SSP1Bでは、Cache7Bには、局所メモリとしてのLocal7Bが含まれているものとする。

[0124] (命令区間記憶部の構成)

図1は、本実施形態における命令区間記憶部2によって実現される再利用表を示している。同図に示すように、命令区間記憶部2は、RB、RF、RO1(第2出力パターン記憶手段)、およびRO2(第1出力パターン記憶手段)を備えた構成となっている。

[0125] RBは、比較すべき値であるレジスタ値または主記憶入力値を格納するValue(値格納領域)、およびキー番号を格納するKey(キー格納領域)を備えており、ValueおよびKeyの組み合わせのラインを複数備えている。

[0126] RFは、次に比較すべきレジスタ番号または主記憶アドレスがないことを示す終端フラグE、次に比較すべきレジスタ番号または主記憶アドレスの内容が更新されたことを示す比較要フラグC、次に比較すべき対象がレジスタか主記憶かを示すR/M、次に比較すべきレジスタ番号または主記憶アドレスを示すAdr.(検索項目指定領域)、直前に参照したライン番号を示すUP(親ノード格納領域)、次に比較すべきレジスタ番号または主記憶アドレスよりも優先して比較すべきレジスタ番号または主記憶アドレスを示すAlt.(比較要項目指定領域)、および、優先して比較する際に必要なキーを示すDN(比較要キー指定領域)を備えており、これらはRBにおける各ラインに対応して設けられている。

[0127] RO1およびRO2は、RBおよびRFによる検索結果により、再利用が可能であると判

定された場合に、主記憶および／またはレジスタに出力する出力値を格納するものである。RO1は、RFの各ラインに1対1で対応して出力値および出力すべきアドレスを格納している。RO2は、RO1のみでは出力値を格納しきれない場合に、格納しきれない分の出力値および出力すべきアドレスを格納している。RO2からも出力値を読み出す必要がある場合には、RO1における該当ラインに、RO2における出力値が格納されているポインタが示されており、このポインタを用いてRO2から出力値の読み出しが行われる。

[0128] また、RBおよびRFは、それぞれCAM(content-addressable memory)およびRAM(Random Access Memory)によって構成されている。一般的に、アドレスが与えられると、そのアドレスに格納された値を参照することができるメモリは、RAMと呼ばれるメモリである。一方、上記のCAMとは、連想メモリと呼ばれるメモリであり、検索すべき内容が与えられると、その内容に一致するラインが選択されるようになっている。通常は、CAMはRAMとセットにして用いられる。

[0129] ここで、CAMとRAMとの連携動作について、具体例を挙げて説明する。CAMに、「5, 5, 5, 5, 5」、「1, 3, 1, 1, 1」、「1, 3, 3, 5, 2」、「6, 6, 6, 6, 6」というデータ列がエントリとして登録されており、RAMに、CAMにおける各データ列に対応して、「5, 5」、「1, 1」、「1, 2」、「6, 6」というデータが登録されているとする。ここで、検索すべきデータ列として、「1, 3, 3, 5, 2」をCAMに入力すると、一致するエントリがONとなり、RAMに登録されている該当するデータ「1, 2」が出力されることになる。この具体例と同様の構成および動作によって、上記RBおよびRFが実現されることになる。

[0130] (比較例)

ここで、比較例として、図8に示すような構成のRFおよびRBによる動作について説明する。同図に示すように、RFは、エントリが有効であるか否かを示す状態表示フラグV、エントリ入れ替えのヒントを示すLRU、関数とループとを区別するF/L、命令区間の先頭アドレスを示すStart、命令区間の終了アドレスを示すEnd、参照すべき主記憶入力アドレスに関する情報を示すRead、および、参照すべき主記憶出力アドレスに関する情報を示すWriteを保持している。

[0131] また、RBは、エントリが有効であるか否かを示す状態表示フラグV、エントリ入れ替えのヒントを示すLRU、命令区間を呼び出す際の直前のスタックポイント%spを示すSP、ループの終了アドレス(End)、ループ終了時の分岐方向を示すtaken/not、レジスタ入力値としての引数(Args.)(V:有効エントリ、Val.:値)および引数以外のレジスタ入力値および条件コード(Regs.,CC)、主記憶入力有効バイトMask、主記憶入力値Value、主記憶出力有効バイトMask、主記憶出力値Value、および、レジスタ出力値としての返り値Return Valuesおよび返り値以外のレジスタ出力値および条件コードRegs.,CC(V:有効エントリ、Val.:値)を保持している。

[0132] 関数またはループを実行する際に、以前に実行した命令区間が再利用可能であるか否かを判断する際には、次の手順で行われる。まず、(1)RFに登録されている関数またはループのエントリの先頭アドレスStartに、該当関数またはループの先頭アドレスと一致するものがあるかを検索する。一致するものがある場合には、(2)RBに登録されている該当エントリのうち、有効エントリを示す状態表示フラグVが登録済状態にセットされているエントリであって、かつ、該エントリにおける引数args.およびRegs.,CCが、呼び出す関数またはループの対応する値と完全に一致するエントリを1つまたは複数選択する。そして、選択したエントリにおいて、(3)関連する主記憶アドレス、すなわち、少なくとも1つのMaskが有効であるReadアドレスを用いて主記憶を順に参照し、(4)該当関数またはループの主記憶入力値と、RBに登録されている主記憶入力値との比較を行う。そして、全ての入力が一致する場合に、(5)RBに記憶されているReturn Valuesをレジスタに書き込み、主記憶出力アドレスに対して、順次、各有効フラグMaskがセットされている主記憶出力値Valueを書き込む。以上により、関数またはループの再利用が実現されることになる。

[0133] 以上のような比較例における動作を、図9を参照しながらより具体的に説明する。まず、プログラムカウンタ(PC)と、RFに登録された命令区間先頭アドレス(Region)とが比較され、さらに、レジスタの内容(Reg.)と、RBに登録されているレジスタ入力値(Args.、Regs.,CC)とを比較する。この時点で、RBにおけるエントリ01ー04のうち、エントリ03およびエントリ04が一致すると判定されたとする。すなわち、この時点では、マルチマッチとなっている。

[0134] 次に、主記憶アドレスA1に関して比較することになるが、主記憶アドレスA1に対しては、RFにおいて、一致比較を行う必要がないことを示すフラグ(0)が示されているので、一致比較は行われない。すなわち、エントリ03およびエントリ04が候補として残ったままとなる。

[0135] 次に、主記憶アドレスA2に関して比較が行われる。ここで、RFにおいて、主記憶アドレスA2に関しては一致比較を行う必要があることを示すフラグ(1)が示されているので、一致比較が行われる。この結果、内容が「00」であるエントリ03のみが候補として残ることになる。その後、一致比較を行う項目として主記憶アドレスA3およびA4があるが、これらはどちらも一致比較を行う必要がないことを示すフラグが示されているので、エントリ03は、比較が必要な全ての項目が一致したことになる。よって、エントリ03に対応する出力値としての主記憶出力値およびレジスタ出力値が主記憶およびレジスタに出力される。

[0136] この比較例における動作のポイントは次の通りである。(a)RBに登録されている各値と再利用対象となっている関数またはループにおける対応する値とを比較する際に、RBにおける縦の列を順に一致確認していくことになるが、内容が一致するエントリが複数存在する(マルチマッチ)ことを許容している。(b)検索途中においてマルチマッチを許容しているが、最終的に1つのエントリが選択されればよい。(c)RBにおける列を一致確認していく順番は任意であるので、例えばレジスタ入力値を最初にまとめて比較する、ということを行うことが可能である。

[0137] また、この比較例の場合、次のような問題がある。(d)RBにおいて、各エントリにおける項目数(横の長さ)は固定となっている。よって、登録されている項目以外の項目を追加することはできないようになっている。また、逆に、使用しない項目に対応するメモリ領域は空き領域となるが、これを有効利用することはできない。(e)各エントリは、1つの項目でも内容が異なれば、それぞれ別のエントリとして登録する必要がある。よって、RBにおけるメモリの利用効率は良くないことになる。

[0138] なお、以上のような比較例の場合、RFおよびRBを構成するメモリとしては、構造が横長のものとなる。例えばこのメモリ容量を2Mbyteとした場合、横が2Kword、縦を256エントリとすることになる。

[0139] （入力パターンを木構造として登録する第1構成例）

上記の比較例では、RBにおける各エントリとしての横の行は、一致比較を行うべき入力値の項目を全て含んだものとなっている。すなわち、全ての入力パターンをそれぞれ1つのエントリとしてRBに登録するようになっている。

[0140] これに対して、本第1構成例では、一致比較を行うべき入力値の項目を短い単位に区切り、それぞれの比較単位をノードとしてとらえ、入力パターンを木構造としてRFおよびRBに登録するようになっている。そして、再利用を行う際には、一致するノードを順次選択することによって、最終的に再利用可能かを判断するようになっている。別の言い方をすれば、複数の入力パターンに共通する部分を1つにまとめて、RFおよびRBの1行に対応づけるようになっている。

[0141] これにより、冗長性をなくし、命令区間記憶部2を構成するメモリの利用効率を向上させることが可能となる。また、入力パターンを木構造としているので、1つの入力パターンをRBにおける1つの行としてのエントリに対応付ける必要がないことになる。よって、一致比較を行うべき入力値の項目の数を可変にすることが可能となっている。

[0142] また、RFおよびRBは、入力パターンを木構造として登録しているので、一致比較を行う際には、マルチマッチが行われないことになる。つまり、命令区間記憶部2としては、シングルマッチ機構を有する連想検索メモリであれば実現可能となる。ここで、シングルマッチ機構のみを有する連想検索メモリは一般的に市販されている一方、マルチマッチをシングルマッチと同一性能によって報告可能な連想検索メモリは一般的には市販されていない。すなわち、本第1構成例における命令区間記憶部2によれば、市販の連想検索メモリを利用することができるので、より短期間かつ低コストで、本実施形態に係るデータ処理装置を実現することが可能となる。

[0143] 次に、図3を参照しながら、上記第1構成例における命令区間記憶部2における連想検索動作の具体例について説明する。まず、命令区間の実行が検出されると、プログラムカウンタ（PC）およびレジスタの内容（Reg.）がRBに入力される。そして、RBにおいて、連想検索により、入力されたこれらの値と、RBのValueの列に登録されている命令区間先頭アドレスおよびレジスタ値とが比較され、値が一致する唯一の行（ライン）が候補（マッチライン）として選択される。この例では、RBにおける「01」のライ

ンがマッチラインとして選択される。

[0144] 次に、マッチラインとして選択されたラインのRBにおける番地である「01」が、エンコード結果としてRFに伝達され、キー01に対応するRFにおけるラインが参照される。キー01に対応するRFにおけるラインでは、比較要フラグが「0」であり、比較すべき主記憶アドレスがA1となっている。すなわち、主記憶アドレスA1に関しては、一致比較を行う必要はないことになる。

[0145] 次に、キー01を用いて、RBにおけるKeyの列に対して検索が行われる。この例では、RBにおける「03」のラインがマッチラインとして選択される。そして、エンコード結果としてキー03がRFに伝達され、キー03に対応するRFにおけるラインが参照される。キー03に対応するRFにおけるラインでは、比較要フラグが「1」であり、比較すべき主記憶アドレスがA2となっている。すなわち、主記憶アドレスA2に関しては、一致比較を行う必要があることになる。ここで、主記憶3における主記憶アドレスA2の値がCache7Aを介して読み出され、RBにおいて、Valueが主記憶3から読み出された値であり、かつ、Keyが「03」となっているラインが検索される。図3に示す例では、Keyが「03」となっているラインは「04」および「05」の2つあるが、主記憶3から読み出された値が「00」であるので、「05」のラインがマッチラインとして選択され、RFに対して、エンコード結果としてキー05が伝達される。

[0146] 以上のような処理が繰り返され、RFにおいて、次に比較すべきレジスタ番号または主記憶アドレスがないことを示す終端フラグEが検出された場合、入力パターンが全て一致したと判定され、該当命令区間は再利用可能と判断される。そして、終端フラグEが検出されたラインから「Select Output」信号が出力され、RO1およびRO2に格納されている、該ラインに対応する出力値がレジスタ6Aおよび主記憶3に対して出力される。

[0147] 以上のように、本第1構成例における命令区間記憶部2による連想検索動作は、次のような特徴を有している。まず、内容が一致したことを示すマッチラインは、RBにおいて1つのラインのみとなるので、検索動作を次列へ伝搬する際にエンコードした結果を1つ伝送すればよいことになる。したがって、RBとRFとの間を接続する信号線は、アドレスのエンコード結果である1組(N本)でよいことになる。これに対して、上記し

た比較例では、RBにおいてマルチマッチが許容されているので、RBにおける各列同士を接続する信号線は、各ラインごとに設ける(2^N本)必要があることになる。すなわち、本第1構成例の構成によれば、命令区間記憶部2を構成する連想検索メモリにおける信号線の数を大幅に低減することが可能となる。

[0148] また、検索途中ではシングルマッチのみが許容されるようになっているので、比較すべき項目の比較順番は、木構造における参照順に限定されることになる。すなわち、レジスタ値とメモリ内容とは、参照順に混在させながら比較する必要がある。

[0149] 入力パターンは、各項目を参照すべきKeyという形でリンクさせることにより、木構造によってRBおよびRFに登録されている。また、入力パターンの項目は、終端フラグによってその終端が示されるようになっている。よって、入力パターンの項目数を可変とすることができるので、再利用表に登録すべき命令区間の状態に応じて、柔軟に入力パターンの項目数を設定することが可能となる。また、入力パターンの項目数が固定でないことによって、利用しない項目が無駄にメモリ領域を占有することがなくなるので、メモリ領域の利用効率を向上させることができる。

[0150] また、木構造によって入力パターンが登録されるので、項目の内容が重複する部分については、複数の入力パターンで1つのラインを共有することが可能となっている。よって、メモリ領域の利用効率をさらに向上させることができる。

[0151] なお、以上のような構成の場合、RFおよびRBを構成するメモリとしては、構造が縦長のものとなる。例えばこのメモリ容量を2Mbyteとした場合、横が8word、縦を65536ラインとすることになる。

[0152] (入力パターンを木構造として登録する第2構成例)

上記の例では、図1に示したRFにおいて、UP、Alt.、およびDNの項目は利用していないことになる。すなわち、上記の例では、RFにおいて、これらの項目を設ける必要はないことになる。これに対して、UP、Alt.、およびDNの項目を利用することによって、連想検索動作をさらに高速化する第2の構成例およびその動作について以下に説明する。

[0153] まず、図4(b)に、プログラムカウンタ(PC)およびレジスタの内容(Reg.)のみを比較し、これらが一致した場合は、主記憶値を比較することなく、区間の再利用が可能で

あると判断できる場合の状態を示す。この状態では、まず、RBの「01」のラインにおいて、PCおよびReg.がValueに登録されており、RFの「01」のラインにおいて、終端フラグが「E」、比較要フラグが「0」、比較すべき主記憶アドレスが「A1」、親ノード番号を示すUPが「FF」となっている。また、RBの「03」のラインでは、Value値なしで、Keyが「01」となっており、RFの「03」のラインでは、終端フラグが「E」、比較要フラグが「0」、比較すべき主記憶アドレスが「A2」、親ノード番号を示すUPが「FF」となっている。以降、同様に、RBおよびRFにおける「05」のラインおよび「07」のラインが登録されており、それぞれ終端フラグが「E」、比較要フラグが「0」となっている。

[0154] この状態で、ある命令区間の実行が検出されると、PCおよびReg.がRBに入力され、マッチラインとして、RBにおける「01」のラインが選択される。そして、マッチラインとして選択されたラインのRBにおける番地である「01」が、エンコード結果としてRFに伝達され、キー01に対応するRFにおけるラインが参照される。キー01に対応するRFにおけるラインでは、終端フラグが「E」となっているので、次に比較すべき主記憶アドレスがないことがわかる。また、比較要フラグ「0」となっているので、主記憶アドレスA1について比較を行う必要はないことがわかる。

[0155] したがって、図4(a)の木構造に示すように、PCおよびReg.の一致がS1において確認されると、Tr1に示すノードのように、主記憶アドレスA1、A2、A3における比較を行うことなく、対応する出力値が出力されることになる。

[0156] RFおよびRBがこの状態である場合に、主記憶アドレスA2に対して書き込みが行われたとする。この場合、RFおよびRBにおける入力パターンの登録時には主記憶アドレスA2の一致比較を行う必要はない状態であったが、主記憶アドレスA2が変更されることによって、主記憶アドレスA2の一致比較を行う必要が生じることになる。したがって、この場合には、図5(b)に示すようにRFおよびRBが変更されることになる。

[0157] まず、内容が変更された主記憶アドレスであるA2をキーにして、RFにおけるAdr.の列に対して検索がかけられる。これによって、RFにおける「03」のラインが選択される。そして、選択された「03」のラインにおいて、比較要フラグが「1」に設定されるとともに、終端フラグ「E」が削除される。

[0158] 次に、「03」のラインにおけるUPを参照することによって、親ノードとしての「01」のラ

インが認識される。そして、「01」のラインにおいて、次に比較すべき主記憶アドレスよりも優先して比較すべき主記憶アドレスを示すAlt.に、内容が変更された主記憶アドレスであるA2が書き込まれるとともに、終端フラグ「E」が削除される。さらに、「01」のラインにおいて、優先して比較する際に必要なキーを示すDNに「03」が書き込まれる。

[0159] 以上のようにRFおよびRBが書き換えられた場合の連想検索動作は次のようになる。ある命令区間が検出された際に、まず、PCおよびReg.がRBに入力される。そして、RBにおいて、連想検索により、入力されたこれらの値と、RBのValueの列に登録されている命令区間先頭アドレスおよびレジスタ値とが比較され、RBにおける「01」のラインがマッチラインとして選択される。

[0160] 次に、マッチラインとして選択されたラインのRBにおける番地である「01」が、エンコード結果としてRFに伝達され、キー01に対応するRFにおけるラインが参照される。キー01に対応するRFにおけるラインでは、比較要フラグが「0」であり、比較すべき主記憶アドレスがA1となっている。すなわち、主記憶アドレスA1に関しては、一致比較を行う必要はないことがわかる。

[0161] また、次に比較すべき主記憶アドレスよりも優先して比較すべき主記憶アドレスを示すAlt.に、主記憶アドレスA2が登録されており、優先して比較する際に必要なキーを示すDNに「03」が登録されていることが確認される。この場合、主記憶3における主記憶アドレスA2の値がCache7Aを介して読み出され、RBにおいて、Valueが主記憶3から読み出された値であり、かつ、Keyが、DNに示されている「03」となっているラインが検索される。

[0162] 図5(b)に示す例では、Keyが「03」となっているラインは「04」および「05」の2つあるが、主記憶3から読み出された値が「00」であるので、「05」のラインがマッチラインとして選択され、RFに対して、エンコード結果としてキー05が伝達される。キー05に対応するRFにおけるラインでは、終端フラグが「E」となっているので、入力パターンが全て一致したと判定され、該当命令区間は再利用可能と判断される。そして、終端フラグEが検出されたラインから「Select Output」信号が出力され、RO1およびRO2に格納されている、該ラインに対応する出力値がレジスタ6Aおよび主記憶3に対して

出力される。

[0163] 以上のような連想検索動作を行う第2の構成例によれば、RFにおいて、次に比較すべき主記憶アドレスよりも優先して比較すべき主記憶アドレスを示すAlt.、および、優先して比較する際に必要なキーを示すDNが設けられているので、図5(a)に示す木構造のように、主記憶アドレスA1の内容とキー01による検索をスキップして、主記憶アドレスA2の内容とキー03による検索が可能となる。したがって、検索動作の処理ステップを低減することができるので、処理の高速化を図ることができる。

[0164] (出力値の格納手段構成例)

上記では、命令区間の入力パターンをRFおよびRBに登録し、連想検索動作を行うことについて説明したが、以下では、入力パターンの一致が確認された後に、再利用として出力される出力値を格納する手段の構成例について説明する。上記において図1を参照しながら説明したように、命令区間記憶部2には、再利用が可能であると判定された場合に、主記憶および/またはレジスタに出力する出力値を格納する出力値格納手段として、RO1およびRO2が設けられている。

[0165] 出力値は、RFおよびRBから出力されるアドレスに基づいて、出力値を記憶するRAMなどの記憶手段を参照することによって得ることが可能である。しかしながら、入力パターンと同様に、出力パターンについても、出力値の項目数を可変とすることが好ましいので、出力値の格納方法に関して工夫が必要である。

[0166] 入力パターンに関しては、RFおよびRBにおいて木構造によって登録されている。そして、木構造の末端となっているライン、すなわち、終端フラグEが登録されているラインにおいて、再利用が可能であると判定されることになる。したがって、終端フラグEが登録されている各ラインに、出力すべき出力値を格納する出力値格納手段におけるポインタを登録しておくことによって、再利用の際の出力動作を行うことが可能となる。

[0167] しかしながら、入力パターンが全て一致したことが確認された時点で、出力値が格納されているポインタに基づいて出力値格納手段における格納位置が特定される場合、ポインタに基づいて格納位置を特定するという変換処理が必要となり、処理速度を低下させる要因となる。

[0168] そこで、本構成例では、出力値格納手段として、RO1およびRO2の2つの記憶手段を設けている。そして、RO1は、RFの各ラインに1対1で対応して出力値および出力すべきアドレスを格納している。すなわち、終端フラグEが登録されているRFのラインにおいて再利用が可能であると判定された場合には、そのラインに対応するRO1のラインが選択され、出力値が出力される。

[0169] しかしながら、このように、出力値格納手段を、RFの各ラインに1対1で対応して出力値および出力すべきアドレスを格納している場合、RFにおける、終端フラグEが登録されていないRFのラインに対しても、RO1においてメモリ領域が確保されることになる。また、終端フラグEが登録されているRFの全てのラインに対応して、RO1において出力値を格納するので、同じ内容が複数箇所で記憶されている、というような冗長性が存在することになる。したがって、RO1は、高速に処理を行うという面では優れているが、メモリの利用効率としてはよくないことになる。

[0170] この問題を解消するために、RO1に登録可能な項目数、すなわち出力値と出力アドレスとの組の数を少なめに設定する(図1の例では2つ)とともに、RO1に登録しきれない出力値および出力アドレスの組については、ポインタを用いて格納領域が指示される構成のRO2に登録するようにしている。

[0171] RO2においては、ポインタによって格納領域が指示されるので、使用されないメモリ領域はほとんど生じないことになる。また、複数の出力値および出力アドレスの組を登録する場合には、順次ポインタを用いてつなげていくことができるので、登録可能な出力値および出力アドレスの組の数を可変にすることが可能である。さらに、RO1における複数のラインから、RO2における同じ格納位置を示すポインタを指示することも可能となるので、RO2における格納情報を、RO1における複数のラインで共有することも可能となる。よって、RO2においては、格納内容の冗長性を低くすることができる。

[0172] 以上のように、出力値格納手段としてRO1およびRO2の2つを設けることによって、出力値の項目が少ない場合にはRO1のみの利用により処理の高速性を実現するとともに、出力値の項目が多い場合には、項目の数を可変とすることが可能なRO2を用いることによって対応している。よって、上記の構成によれば、処理の高速性とメモ

リ利用効率の向上とを実現することができる。

[0173] （命令区間記憶部に対する登録処理）

上記では、ある命令区間の実行に際して再利用を行う場合の動作について説明した。以下では、ある命令区間の実行に際して、再利用が行えないと判断された場合に、該命令区間による入出力をRF、RB、RO1、およびRO2に登録する際の動作について説明する。

[0174] まず、ある命令区間の実行が検出されると、PCおよびReg.の値がRBに入力される。そして、RBにおいて、連想検索により、入力されたこれらの値と、RBのValueの列に登録されている命令区間先頭アドレスおよびレジスタ値とが比較される。ここで、RBのValueの列に、入力された値と一致するものがないと判定された場合、該命令区間は、再利用が不可能であると判定され、演算器5Aによる演算処理が行われる。そして、該当命令区間の演算処理が終了するまでに用いられるレジスタ入力値、主記憶入力値、主記憶出力値、およびレジスタ出力値が、RB、RF、RO1、必要に応じてRO2に登録される。ここで、RBおよびRFに登録を行う際には、上記で示したような木構造となるように、各項目が1つのラインに対応するように登録が行われる。そして、登録すべき入力パターンの最後の項目が登録されたラインにおいて、RFの終端フラグを「E」とし、入力パターンの登録を終了する。

[0175] 一方、入力されたPCおよびReg.の値に一致するものが、RBのValueの列に登録されている場合には、上記した連想検索動作と同様にして、次の一致比較すべき項目についての一致比較が行われる。このようにして、RBおよびRFに登録されている入力パターンと、該当命令区間における入力パターンとの一致比較を継続していき、一致しない項目が生じた時点で、新たにノードを追加する形で、その一致しない項目についてRBおよびRFに登録が行われる。そして、登録すべき入力パターンの最後の項目が登録されたラインにおいて、RFの終端フラグを「E」とし、入力パターンの登録を終了する。

[0176] 入力パターンの登録が終了すると、終端フラグを「E」としたRFにおけるラインに対応する、RO1におけるラインに、出力値および出力アドレスの登録を行う。そして、出力値として登録すべき項目がRO1に登録しきれない場合には、ポインタを用いてRO

2に対して登録が行われる。以上により、命令区間の登録処理が完了する。

[0177] (命令区間実行時の入出力セットの生成)

ある命令区間を実行した際に、命令区間記憶部2に対して実行結果が登録されることになるが、この実行結果は、該命令区間の実行に際して、レジスタおよび／または主記憶(以降、単にレジスタ／メモリと称する)に対して行われた入出力のセットに相当するものである。以下では、命令区間記憶部2に登録すべき入出力セットをどのように生成するかについて説明する。

[0178] 上記した入力パターンを木構造として登録する第1および第2構成例の場合、入出力セットはRW4A・4Bによって生成され、生成された入出力セットに基づいて、RB、RF、RO1、およびRO2への上記したような登録処理が行われる。RW4A・4Bは、ある命令区間が実行された際に行われるレジスタ／メモリからの読み出し、および／または、レジスタ／メモリへの書き込みを監視し、これに基づいて入出力セットを生成する。このRW4A・4Bによる入出力セットの生成方法について以下に説明する。なお、以下の説明では、RW4Aについて説明するが、RW4Bについても同様である。

[0179] (RWの第1構成例その1)

図12は、第1構成例その1としてのRW4Aのメモリ構成の概略を示す図である。同図に示すように、RW4Aは、命令区間のPC値を格納するPC、入力アドレスおよび入力値を格納するRWI、および、出力アドレスおよび出力値を格納するRWOのメモリを有している。ある命令区間を実行した際の入出力セットはこのRW4Aのメモリに格納され、その後、命令区間記憶部2に登録されることになる。

[0180] まず、ある命令区間の実行が開始されると、そのPC値がRW4AにおけるPCに格納される。その後、命令区間の実行が順次行われると、レジスタ／メモリからの読み出し、および／または、レジスタ／メモリへの書き込みが順に行われることになる。

[0181] 命令区間実行時にレジスタ／メモリからの読み出しが行われた場合には、RW4Aによって次の処理が行われる。

[0182] (AR1)読み出しが行われたレジスタ／メモリのアドレスが、RWOに登録されているか否かが検索される。RWOに登録されている場合には、既に出力値として入出力セットに登録されている値の読み出しが行われたものであるので、入力値として登録す

る必要はないことになる。すなわち、該アドレスをRWIに登録せずに終了する。

[0183] (AR2)読み出しが行われたレジスタ／メモリのアドレスが、RWOに登録されていない場合には、該アドレスがRWIに登録されているか否かが検索される。RWIに登録されている場合には、既に入力値として入出力セットに登録されている値の読み出しが行われたものであるので、さらに入力値として登録する必要はないことになる。すなわち、該アドレスをRWIに登録せずに終了する。

[0184] (AR3)読み出しが行われたレジスタ／メモリのアドレスが、RWOおよびRWIのいずれにも登録されていない場合には、該アドレスおよび値を入力アドレスおよび入力値としてRWIに登録する。

[0185] また、命令区間実行時にレジスタ／メモリへの書き込みが行われた場合には、RW4Aによって次の処理が行われる。

[0186] (AW1)書き込みが行われたレジスタ／メモリのアドレスが、RWOに登録されているか否かが検索される。RWOに登録されている場合には、既に出力値として入出力セットに登録されている値の書き換えが行われたことになるので、登録されている出力アドレスに対応する出力値を、書き込みが行われた値に更新し、終了する。

[0187] (AW2)書き込みが行われたレジスタ／メモリのアドレスが、RWOに登録されていない場合には、該アドレスおよび値を出力アドレスおよび出力値としてRWOに登録する。

[0188] 以上の処理が該命令区間の終了まで行われることによって、該命令区間の入出力セットがRW4Aによって生成されることになる。生成された入出力セットは、上記したような登録処理によって命令区間記憶部2に登録される。

[0189] ここで、命令区間の一例として、図11に示す命令区間を実行した場合の例について説明する。同図において、PCは、該命令区間が開始された際のPC値を示している。このPC値が、RW4AのPCに格納される。

[0190] その後、第1行目において、レジスタにおけるアドレスR1に格納されている(00001000)という値が読み込まれるとともに、この読み込まれた値に100を加える演算が行われた結果の主記憶アドレス(アドレスA1に相当)の値を読み出す命令が行われている。この時点では、アドレスR1はRWOおよびRWIのいずれにも登録されてい

ないので、アドレスR1および値(00001000)がRWIに登録される。また、アドレスA1の値(----FF--)が読み出され、レジスタのアドレスreg. に格納する命令が行われている。この時点では、アドレスA1はRWOおよびRWIのいずれにも登録されていないので、アドレスA1および値(----FF--)がRWIに登録される。

[0191] また、この時点では、アドレスreg. はRWOに登録されていないので、アドレスreg. および値(----FF--)がRWOに登録される。

[0192] 次に、第2行目において、アドレスreg. から値を読み出して主記憶への書き込み処理が行われ、アドレスB1に値(----FF--)が書き込まれる。この時点では、アドレスreg. はRWOに登録されているので、RWOへの登録は行われない。また、アドレスB1はRWOに登録されていないので、アドレスB1および値(----FF--)がRWOに登録される。

[0193] 次に、第3行目において、レジスタにおけるアドレスR1に格納されている(00001000)という値が読み込まれるとともに、この読み込まれた値に200を加える演算が行われた結果の主記憶アドレス(アドレスA2に相当)の値を読み出す命令が行われている。この時点では、アドレスR1はRWIに既に登録されているので、RWIへの登録は行われない。また、アドレスA2の値(--01----)が読み出され、レジスタのアドレスreg. に格納する命令が行われている。この時点では、アドレスA2はRWOおよびRWIのいずれにも登録されていないので、アドレスA2および値(--01----)がRWIに登録される。

[0194] また、この時点では、アドレスreg. はRWOに登録されており、このRWOにおけるアドレスreg. の値が値(--01----)に更新される。

[0195] 次に、第4行目において、アドレスreg. から値を読み出して主記憶への書き込み処理が行われ、アドレスB2に値(--01----)が書き込まれる。この時点では、アドレスreg. はRWOに登録されているので、RWOへの登録は行われない。また、アドレスB2はRWOに登録されていないので、アドレスB2および値(--01----)がRWOに登録される。

[0196] 次に、第5行目において、アドレスA3の値(5678----)が読み出され、レジスタのアドレスreg. に格納する命令が行われている。この時点では、アドレスA3はRWOおよ

びRWIのいずれにも登録されていないので、アドレスA3および値(5678——)がRWIに登録される。

[0197] また、この時点では、アドレスreg. はRWOに登録されており、このRWOにおけるアドレスreg. の値が値(5678——)に更新される。

[0198] 最後に、第6行目において、アドレスreg. から値を読み出して主記憶への書き込み処理が行われ、アドレスB3に値(5678——)が書き込まれる。この時点では、アドレスreg. はRWOに登録されているので、RWOへの登録は行われない。また、アドレスB3はRWOに登録されていないので、アドレスB3および値(5678——)がRWOに登録される。以上の処理によって、図12に示すRW4Aの入出力セットが生成される。

[0199] 以上のようにして生成された入出力セットは、図13に示すような木構造として、命令区間記憶部2に登録される。この木構造において、登録されている入力パターンは、ルートノードからリーフへ至る1本のパスとして命令区間記憶部2に保持される。以降、命令区間を実行する前に、該命令区間の入力パターンが、登録されている入力パターンと同じであるかを判断するために、図3に示したように、ルートノードから順に、ノードに記録されているアドレスを参照し、得られた値と一致するノードを連想検索機構を用いて選択することを繰り返すことになる。

[0200] (RWの第1構成例その2)

図16は、第1構成例その2としてのRW4Aのメモリ構成の概略を示す図である。同図に示すように、RW4Aは、命令区間のPC値を格納するPC、入力アドレスおよび入力値を格納するRWI、および、出力アドレスおよび出力値を格納するRWOのメモリを有している。ある命令区間を実行した際の入出力セットはこのRW4Aのメモリに格納され、その後、命令区間記憶部2に登録されることになる。

[0201] まず、ある命令区間の実行が開始されると、そのPC値がRW4AにおけるPCに格納される。その後、命令区間の実行が順次行われると、レジスタ/メモリからの読み出し、および/または、レジスタ/メモリへの書き込みが順に行われることになる。

[0202] 命令区間実行時にレジスタ/メモリからの読み出しが行われた場合には、RW4Aによって前記した(AR1)、(AR2)、(AR3)の処理が行われる。また、命令区間実行時にレジスタ/メモリへの書き込みが行われた場合には、RW4Aによって前記した(

AW1)、(AW2)の処理が行われる。

[0203] 以上の処理が該命令区間の終了まで行われることによって、該命令区間の入出力セットがRW4Aによって生成されることになる。生成された入出力セットは、上記したような登録処理によって命令区間記憶部2に登録される。

[0204] ここで、命令区間の一例として、図15に示す命令区間を実行した場合の例について説明する。同図において、PCは、該命令区間が開始された際のPC値を示している。このPC値が、RW4AのPCに格納される。

[0205] 第1行目の命令(以降、単に第1の命令のように称する)において、メモリにおけるアドレスA1からロードした4バイトデータ(00110000)が、レジスタにおけるアドレスR1に格納される。この時点では、読み出しが行われたアドレスA1は、RWOおよびRWIのいずれにも登録されていないので、アドレスA1およびデータ(00110000)がRWIに登録される。また、書き込みが行われたアドレスR1は、RWOに登録されていないので、アドレスR1およびデータ(00110000)がRWOに登録される。

[0206] 次に、第2の命令において、メモリにおけるアドレスA2からロードした1バイトデータ(02)が、レジスタにおけるアドレスR2に格納される。この時点では、読み出しが行われたアドレスA2は、RWOおよびRWIのいずれにも登録されていないので、アドレスA2およびデータ(02)がRWIに登録される。この際に、アドレスA2における残り3バイトについては、Don't Careを意味する「-」が格納される。また、書き込みが行われたアドレスR2は、RWOに登録されていないので、アドレスR2およびデータ(02)がRWOに登録される。

[0207] 次に、第3の命令において、メモリにおけるアドレス(A2+R2)からロードした1バイトデータ(22)が、レジスタにおけるアドレスR2に格納される。ここで、アドレスR2に格納されているデータは(02)であったので、読み出しが行われたメモリにおけるアドレスは(A2+02)となる。この時点では、読み出しが行われたアドレス(A2+02)は、RWOおよびRWIのいずれにも登録されていないので、アドレス(A2+02)およびデータ(22)がRWIに登録される。この際に、アドレスA2における4バイトのうち、アドレス(A2+02)となるバイトの部分にデータ(22)が登録される。すなわち、第2の命令において、アドレスA2となるバイトの部分にデータ(02)が登録されているので、アドレス(

A2+01)およびアドレス(A2+03)となるバイトの部分に、Don't Careを意味する「-」が格納されたままとなる。

[0208] また、書き込みが行われたアドレスR2は、既にRWOに登録されているので、アドレスR2に対応する出力値として、データ(02)からデータ(22)に書き換えられる。

[0209] 次に、第4の命令において、メモリにおけるアドレスA3からロードした1バイトデータ(33)が、レジスタにおけるアドレスR3に格納される。この時点では、読み出しが行われたアドレスA3は、RWOおよびRWIのいずれにも登録されていないので、アドレスA1およびデータ(33)がRWIに登録される。また、書き込みが行われたアドレスR3は、RWOに登録されていないので、アドレスR3およびデータ(33)がRWOに登録される。

[0210] 最後に、第5の命令において、メモリにおけるアドレス(R1+R2)からロードした1バイトデータ(44)が、レジスタにおけるアドレスR4に格納される。ここで、アドレスR1およびR2は、命令区間の内部にて上書きされたレジスタであるので、命令区間の入力とはならない。一方、(R1+R2)によって生成されたアドレスA4は命令区間の入力となる。このアドレスA4は、RWOおよびRWIのいずれにも登録されていないので、アドレスA4およびデータ(44)がRWIに登録される。また、書き込みが行われたアドレスR4は、RWOに登録されていないので、アドレスR4およびデータ(44)がRWOに登録される。以上の処理によって、図16に示すRW4Aの入出力セットが生成される。

[0211] 以上のようにして生成された入出力セットは、図17に示すような木構造として、命令区間記憶部2に登録される。この木構造において、登録されている入力パターンは、ルートノードからリーフへ至る1本のパスとして命令区間記憶部2に保持される。以降、命令区間を実行する前に、該命令区間の入力パターンが、登録されている入力パターンと同じであるかを判断するために、図3に示したように、ルートノードから順に、ノードに記録されているアドレスを参照し、得られた値と一致するノードを連想検索機構を用いて選択することを繰り返すことになる。

[0212] (木構造連想検索の問題)

上記の木構造の場合、入力パターンを1つずつ順に読み出して連想検索を行い、一致するノードが見つかった後に、次のノードの選択を行うことになる。すなわち、先

行するノードの検索が完全に終了してから次のノードの検索が開始されることになる。

[0213] ここで、CAM／RAMで構成される連想検索装置は、一般的に長レイテンシ高スループットの特性を有している。すなわち、一般的な連想検索装置は、1つの検索入力が行われてから出力されるまでの期間は比較的長いものであるが、複数の検索入力を同時に処理して出力することが可能であるという特性を有している。これに対し、上記のように、先行するノードの検索が完全に終了してから次のノードの検索が開始される、というような検索が行われる場合、連想検索装置における高スループットの能力を利用することができないことになり、連想検索装置の能力を十全に発揮することができないことになる(問題1)。

[0214] また、上記の木構造の場合、命令区間の入力パターンが参照順に一本のパスとして実現されており、入力パターン全体が一致しなければ出力を再利用することができないことになる。ここで、次のような例を想定する。まず、ある命令区間を実行した際の入力パターンのうち、前半がパターンA1、後半がパターンA2となっており、パターンA1に対応する出力がX1、パターンA2に対応する出力がX2となっていたとする。また、別の命令区間を実行した際の入力パターンのうち、前半がパターンB1、後半がパターンB2となっており、パターンB1に対応する出力がY1、パターンB2に対応する出力がY2となっていたとする。その後、ある命令区間を実行しようとした時の入力パターンのうち、前半がパターンA1、後半がパターンB2となっていた場合、入力パターンの前半および後半のそれぞれについては再利用が可能であるものの、入力パターン全体としては過去に同一パターンが出現していないので、実際には再利用することができないことになる(問題2)。

[0215] 例えば、図16に示す例では、入力セットにおけるアドレスA1およびA2による入力パターンと、アドレスA3による入力パターンとは、相互に依存関係がなく、互いに独立となっている。すなわち、アドレスA1およびA2による入力パターンを含むが、アドレスA3による入力パターンを含まない命令区間や、アドレスA3による入力パターンを含むが、アドレスA1およびA2による入力パターンを含まない命令区間に関しては、図16に示す入力パターンの木構造が存在したとしても、再利用することができない

ことになる。

[0216] (RWの第2構成例)

上記の2つの問題は、ある呼び出し時点における命令区間の入力パターンをルートノードからリーフへ至る1本のパスによる表現したことによって生じたものである。これらの問題を解決するためには、入力パターンをグループ分割し、各グループ毎に過去の入力パターンを保持する木構造を構成し、さらに、複数木構造の同時探索を可能とすることによって連想検索装置を有効に利用できるようにすることが必要である。例えば、図13に示すような木構造を、図14に示すような複数の木構造に分割して、ルートノードからリーフに至るパスに対応する入力グループ毎に独立に再利用が行われるようにすればよい。

[0217] また、例えば、図16に示す入出力セットに対して、図18に示すように、互いに独立な入力セットおよび出力セットそれぞれにグループ番号を付与する。すなわち、アドレスA1による入力パターン、アドレスA2による入力パターン、およびアドレスA4による入力パターンにグループ番号(grpid)0を付与し、アドレスA3による入力パターンにグループ番号(grpid)2を付与する。また、アドレスR1による出力パターン、アドレスR2による出力パターン、およびアドレスR4による出力パターンにグループ番号(grpid)0を付与し、アドレスR3による出力パターンにグループ番号(grpid)2を付与する。

[0218] 次に、グループ番号に基づいて、図17に示す木構造を、図19に示すような複数の木構造に分割する。このようにすれば、グループの異なる入力パターンを独立に登録することが可能となり、ルートノードからリーフに至るパスに対応する入力グループ毎に独立に再利用を行うことが可能となるとともに、並列に検索を行うことが可能となる。

[0219] 上記のように、木構造の分割を実現するためには、各入力グループ同士の間でデータ依存関係がないことが必要である。すなわち、ある入力パターンをグループAとグループBとに分割した場合において、グループAの入力がグループBの入力に依存する場合、あるいは、グループBの入力がグループAの入力に依存する場合には、グループ分割したとしても、各グループを独立に再利用できる可能性は極めて低くなる。

[0220] データ依存関係がないグループに分割するには、入力パターンを生成する際に、データ依存関係の解析を行う必要がある。すなわち、RW4Aが、データ依存関係の解析を行った上で、入力パターンをデータ依存関係がないグループに分割して入出力セットを生成するようにすればよいことになる。

[0221] 図10は、上記を実現する第2構成例としてのRW4Aの概略構成を示している。同図に示すように、RW4Aは、命令区間のPC値を格納するPC、入力アドレスおよび入力値を格納するRWI、出力アドレスおよび出力値を格納するRWO、依存関係格納部M、行間論理積比較部(入出力グループ設定手段)MR、およびグループID格納部IDを有している。

[0222] 依存関係格納部Mは、2次元配列のメモリであり、各メモリ要素には0または1が記憶されるようになっている。また、依存関係格納部Mにおいて、各列はRWIに登録されている各入力アドレスおよび入力値に対応しており、各行はRWOに登録されている各出力アドレスおよび出力値に対応している。そして、依存関係格納部Mは、各出力アドレスおよび出力値が、どの入力アドレスおよび入力値を起源とするものであるかを示している。

[0223] 行間論理積比較部MRは、依存関係格納部Mに格納されている各行成分間の論理積演算を行い、1以上の出力アドレスおよび出力値を含む出力パターンと、1以上の入力アドレスおよび入力値を含む入力パターンとからなる入出力グループを設定する演算部である。この行間論理積比較部MRによる論理積演算の詳細については後述する。

[0224] グループID格納部IDは、行間論理積比較部MRによる論理積演算結果に基づいて、依存関係格納部Mにおける各列に対応する入力アドレスおよび入力値に対して付与されるグループIDを格納するメモリである。このグループIDの詳細については後述する。

[0225] ある命令区間の実行が開始されると、まず依存関係格納部Mにおける各メモリ要素の初期値として、全て0に設定される。そして、該命令区間のPC値がRW4AにおけるPCに格納される。その後、命令区間の実行が順次行われると、レジスタ/メモリからの読み出し、および/または、レジスタ/メモリへの書き込みが順に行われることに

なる。

[0226] 命令区間実行時にレジスタ／メモリからの読み出しが行われた場合には、RW4Aによって次の処理が行われる。

[0227] (BR1)読み出しが行われたレジスタ／メモリのアドレスが、RWOに登録されているか否かが検索される。RWOに登録されている場合には、既に出力値として入出力セットに登録されている値の読み出しが行われたものであるので、入力値として登録する必要はないことになる。すなわち、該アドレスをRWIに登録せずに終了する。

[0228] この時、RWOにおいて既に登録されているアドレスに対応する依存関係格納部Mの行成分の各メモリ要素の値が取り出され、行成分のみの1次元行列としての暫定行列A(x)として記憶される。ここで、xは暫定行列Aが生成された順に付される番号とする。この暫定行列A(x)は、後述する書き込み処理が終了した時点で初期化される。なお、この暫定行列A(x)は、図10では図示していないが、暫定行列A(x)を複数格納することができる暫定行列格納メモリに格納されることになる。

[0229] (BR2)読み出しが行われたレジスタ／メモリのアドレスが、RWOに登録されていない場合には、該アドレスがRWIに登録されているか否かが検索される。RWIに登録されている場合には、既に入力値として入出力セットに登録されている値の読み出しが行われたものであるので、さらに入力値として登録する必要はないことになる。すなわち、該アドレスをRWIに登録せずに終了する。

[0230] この時、RWIにおいて既に登録されているアドレスに対応する依存関係格納部Mの列に対応するメモリ要素を1とし、その他のメモリ要素を0とした暫定行列A(x)が記憶される。

[0231] (BR3)読み出しが行われたレジスタ／メモリのアドレスが、RWOおよびRWIのいずれにも登録されていない場合には、該アドレスおよび値を入力アドレスおよび入力値としてRWIに登録する。

[0232] この時、新たに追加した入力アドレスおよび入力値(エントリ)に対応する依存関係格納部Mの列に対応するメモリ要素を1とし、その他のメモリ要素を0とした暫定行列A(x)が記憶される。

[0233] また、命令区間実行時にレジスタ／メモリへの書き込みが行われた場合には、RW

4Aによって次の処理が行われる。

[0234] (BW1)書き込みが行われたレジスタ／メモリのアドレスが、RWOに登録されているか否かが検索される。RWOに登録されている場合には、既に出力値として入出力セットに登録されている値の書き換えが行われたことになるので、登録されている出力アドレスに対応する出力値を、書き込みが行われた値に更新し、終了する。

[0235] この時、RWOにおいて既に登録されているアドレスに対応する依存関係格納部Mの行成分が、その時点で記憶されている全ての暫定行列A(x)の論理和に置き換えられる。これにより、RWOにおいて既に登録されている出力アドレス／値に対する出力の起源となる入力アドレス／値のパターンが、該出力アドレスに対応する依存関係格納部Mの行成分によって示されることになる。書き込み処理が終了し、暫定行列A(x)の論理和への置き換えが完了すると、暫定行列A(x)が全て初期化される。

[0236] (BW2)書き込みが行われたレジスタ／メモリのアドレスが、RWOに登録されていない場合には、該アドレスおよび値を出力アドレスおよび出力値としてRWOに登録する。

[0237] この時、新たに追加した出力アドレスおよび出力値(エントリ)に対応する依存関係格納部Mの行成分が、その時点で記憶されている全ての暫定行列A(x)の論理和に置き換えられる。これにより、RWOに新たに登録した出力アドレス／値に対する出力の起源となる入力アドレス／値のパターンが、該出力アドレスに対応する依存関係格納部Mの行成分によって示されることになる。書き込み処理が終了し、暫定行列A(x)の論理和への置き換えが完了すると、暫定行列A(x)が全て初期化される。

[0238] ここで、命令区間の一例として、図11に示す命令区間を実行した場合の例について説明する。同図において、PCは、該命令区間が開始された際のPC値を示している。このPC値が、RW4APCに格納される。

[0239] その後、第1行目において、レジスタにおけるアドレスR1に格納されている(00001000)という値が読み込まれるとともに、この読み込まれた値に100を加える演算が行われた結果の主記憶アドレス(アドレスA1に相当)の値を読み出す命令が行われている。この時点では、アドレスR1はRWOおよびRWIのいずれにも登録されていないので、アドレスR1および値(00001000)がRWIに登録される。

[0240] この時、アドレスR1に対応する依存関係格納部Mの列に対応するメモリ要素を1とし、その他のメモリ要素を0とした暫定行列A(1)[1000]が記憶される。

[0241] また、アドレスA1の値(——FF—)が読み出され、レジスタのアドレスreg. に格納する命令が行われている。この時点では、アドレスA1はRWOおよびRWIのいずれにも登録されていないので、アドレスA1および値(----FF--)がRWIに登録される。

[0242] この時、アドレスA1に対応する依存関係格納部Mの列に対応するメモリ要素を1とし、その他のメモリ要素を0とした暫定行列A(2)[0100]が記憶される。

[0243] また、この時点では、アドレスreg. はRWOに登録されていないので、アドレスreg. および値(——FF—)がRWOに登録される。この時、新たに追加したアドレスreg. に対応する依存関係格納部Mの行成分が、その時点で記憶されている全ての暫定行列A(1)ーA(2)の論理和[1100]に置き換えられる。その後、暫定行列A(x)が初期化される。

[0244] 次に、第2行目において、アドレスreg. から値を読み出して主記憶への書き込み処理が行われ、アドレスB1に値(----FF—)が書き込まれる。この時点では、アドレスreg. はRWOに登録されているので、RWOへの登録は行われない。この時、アドレスreg. に対応する依存関係格納部Mの行成分が取り出され、暫定行列A(1)[1100]が記憶される。

[0245] また、アドレスB1はRWOに登録されていないので、アドレスB1および値(——FF—)がRWOに登録される。

[0246] この時、新たに追加した出力アドレスに対応する依存関係格納部Mの行成分が、その時点で記憶されている全ての暫定行列A(1)の論理和[1100]に置き換えられる。その後、暫定行列A(x)が初期化される。

[0247] 次に、第3行目において、レジスタにおけるアドレスR1に格納されている(00001000)という値が読み込まれるとともに、この読み込まれた値に200を加える演算が行われた結果の主記憶アドレス(アドレスA2に相当)の値を読み出す命令が行われている。この時点では、アドレスR1はRWIに既に登録されているので、RWIへの登録は行われない。

[0248] この時、アドレスR1に対応する依存関係格納部Mの列に対応するメモリ要素を1と

し、その他のメモリ要素を0とした暫定行列A(1)[1000]が記憶される。

[0249] また、アドレスA2の値(--01----)が読み出され、レジスタのアドレスreg. に格納する命令が行われている。この時点では、アドレスA2はRWOおよびRWIのいずれにも登録されていないので、アドレスA2および値(—01——)がRWIに登録される。

[0250] この時、アドレスA2に対応する依存関係格納部Mの列に対応するメモリ要素を1とし、その他のメモリ要素を0とした暫定行列A(2)[0010]が記憶される。

[0251] また、この時点では、アドレスreg. はRWOに登録されており、このRWOにおけるアドレスreg. の値が値(—01———)に更新される。この時、更新されたアドレスreg. に対応する依存関係格納部Mの行成分が、その時点で記憶されている全ての暫定行列A(1)ーA(2)の論理和[1010]に置き換えられる。その後、暫定行列A(x)が初期化される。

[0252] 次に、第4行目において、アドレスreg. から値を読み出して主記憶への書き込み処理が行われ、アドレスB2に値(—01——)が書き込まれる。この時点では、アドレスreg. はRWOに登録されているので、RWOへの登録は行われない。この時、アドレスreg. に対応する依存関係格納部Mの行成分が取り出され、暫定行列A(1)[1010]が記憶される。

[0253] また、アドレスB2はRWOに登録されていないので、アドレスB2および値(—01——)がRWOに登録される。

[0254] この時、新たに追加した出力アドレスに対応する依存関係格納部Mの行成分が、その時点で記憶されている全ての暫定行列A(1)の論理和[1010]に置き換えられる。その後、暫定行列A(x)が初期化される。

[0255] 次に、第5行目において、アドレスA3の値(5678----)が読み出され、レジスタのアドレスreg. に格納する命令が行われている。この時点では、アドレスA3はRWOおよびRWIのいずれにも登録されていないので、アドレスA3および値(5678——)がRWIに登録される。

[0256] この時、アドレスA3に対応する依存関係格納部Mの列に対応するメモリ要素を1とし、その他のメモリ要素を0とした暫定行列A(1)[0001]が記憶される。

[0257] また、この時点では、アドレスreg. はRWOに登録されており、このRWOにおけるア

ドレスreg. の値が値(5678——)に更新される。この時、更新されたアドレスreg. に対応する依存関係格納部Mの行成分が、その時点で記憶されている全ての暫定行列A(1)の論理和[0001]に置き換えられる。その後、暫定行列A(x)が初期化される。

[0258] 最後に、第6行目において、アドレスreg. から値を読み出して主記憶への書き込み処理が行われ、アドレスB3に値(5678——)が書き込まれる。この時点では、アドレスreg. はRWOに登録されているので、RWOへの登録は行われない。この時、アドレスreg. に対応する依存関係格納部Mの行成分が取り出され、暫定行列A(1)[0001]が記憶される。

[0259] また、アドレスB3はRWOに登録されていないので、アドレスB3および値(5678——)がRWOに登録される。

[0260] この時、新たに追加した出力アドレスに対応する依存関係格納部Mの行成分が、その時点で記憶されている全ての暫定行列A(1)の論理和[0001]に置き換えられる。その後、暫定行列A(x)が初期化される。以上の処理によって、図10に示すRW4Aの入出力セットが生成される。

[0261] 以上のように依存関係格納部Mを生成することによって、命令区間の実行完了時には、次の情報が得られていることになる。

[0262] (Rs1)依存関係格納部Mの行成分は、対応する出力アドレス／値の起源となる入力アドレス／値を1によって示している。

[0263] (Rs2)ある行成分Maにおいて1が示されている入力アドレス／値の組が1つの入力グループを形成し、該入力グループが一致した場合に再利用可能な出力アドレス／値は、行成分Maに対応する出力アドレス／値である。

[0264] (Rs3)「ある行成分Maの反転」と「ある行成分Mb」との論理積が全て0である場合、Maにおける1のパターンは、Mbにおける1のパターンを包含する。すなわち、Maに属する入力アドレス／値の組が1つの入力グループを形成するとともに、該入力グループが一致した場合に再利用可能な出力アドレス／値は、Maに対応する出力アドレス／値、および、Mbに対応する出力アドレス／値となる。

[0265] (Rs4)「ある行成分Ma」と「ある行成分Mb」との論理積が全て0である場合、Maに

属する入力アドレス／値と、Mbに属する入力アドレス／値とは互いに独立している。

[0266] 以上の情報に基づいて、RW4Aは、入出力セットを複数の入出力グループに分割する。まず、上記の(Rs3)に関連する処理として、依存関係格納部Mにおいて、「ある行成分Maの反転」と「ある行成分Mb」との論理積が全て0になる行成分の組が行間論理積比較部MRによって抽出される。抽出された行成分の組のうち、入力アドレス／値の組を最も多く含む行成分、すなわち、他の行成分における入力アドレス／値の組を全て含んだ行成分が上位行成分として選択される。そして、抽出された行成分のうち、上位行成分以外の下位行成分が削除される。この処理によって、冗長な入出力グループを排除することができる。

[0267] 次に、下位行成分が削除された状態において、上記の(Rs4)に関連する処理として、「ある行成分Ma」と「ある行成分Mb」との論理積が全て0になる行成分の組が行間論理積比較部MRによって抽出される。そして、抽出された行成分の組のうち、他のどの行成分に対しても論理積が全て0になる行成分がさらに抽出される。ここで抽出された行成分は、他のどの行成分に対しても依存関係を有さないことになるので、これを独立行成分と設定し、これ以外を非独立行成分と設定する。

[0268] 独立行成分は、それぞれ対応する入力アドレス／値の組および出力アドレス／値の組が抽出されて、1つの入出力グループとして設定される。一方、非独立行成分は、次の2つの処理のいずれかによって入出力グループとして設定される。

[0269] 第1の処理としては、非独立行成分の全てに含まれる入力アドレス／値の組および出力アドレス／値の組の総和を1つの入出力グループとして設定する処理である。第2の処理としては、非独立行成分のそれぞれをそのまま入出力グループとして設定する処理である。第1の処理を行う場合、入出力グループの数を必要以上に増大させることがなくなるので、命令区間記憶部2におけるメモリ使用容量を低減することができる。一方、第2の処理を行う場合、入出力グループの数が比較的多くなり、命令区間記憶部2におけるメモリ使用容量が比較的大きくなるという問題はあるが、命令区間記憶部2において、同時に検索すべき木構造の数を増やすことができるので、連想検索装置における高スループットの能力を利用することが可能となる。

[0270] 以上のようにして入出力グループが設定されると、これに基づいて、行間論理積比

較部MRが、各入出力グループにグループIDを付与し、RWIに登録されている入力アドレス／値のそれぞれに対して、どのグループIDに含まれているものであるかを示す情報をグループID格納部IDに格納する。これにより、グループID格納部IDの内容を見ることによって、各入出力グループにおける入力パターンを特定することが可能となる。

[0271] 以上のように、RW4Aは、1つ以上の入出力グループを生成し、生成した入出力グループを命令区間記憶部2に対して実行結果として登録する。このような処理により、1つの命令区間の実行結果が、1つ以上の入出力グループとして命令区間記憶部2に登録されることになる。よって、ある命令区間を再利用によって実行する際に、以前に実行された命令区間の入力パターンの一部しか一致していない場合でも、再利用を行うことが可能となる確率を高めることができる。また、同時に検索すべき木構造が複数存在する確率を高めることができるので、連想検索装置における高スループットの能力を利用することが可能となり、処理速度の向上を期待することができる。

[0272] なお、本実施形態においては、RW4Aによって生成された入出力グループは、入力パターンを木構造として登録する命令区間記憶部2に登録されるようになっているが、これに限定されるものではない。すなわち、RW4Aによって生成された入出力グループを、命令区間の実行結果を再利用することが可能な形態で登録することが可能な命令区間記憶部であれば、本実施形態に係るRW4Aを適用することが可能である。

[0273] （RWの第3構成例）

上記のRWの第2構成例によれば、依存関係格納部Mを用いて行間論理積比較部MRが演算を行うことによって、1つの命令区間の実行結果を、1つ以上の入出力グループとして命令区間記憶部2に登録することが可能となる。しかしながら、命令区間の実行結果において、出力アドレスおよび出力値のパターンの数が多くなると、依存関係格納部Mの行数が多くなることになる。この場合、行間論理積比較部MRによる論理積演算が膨大になり、行間論理積比較部MRの回路規模が莫大になるという問題がある。以下に示すRWの第3構成例は、この問題を解決するものとなっている。

[0274] 図20は、第3構成例としてのRW4Aの概略構成を示している。同図に示すように、RW4Aは、命令区間の命令区間のPC値を格納するPC、入力アドレスおよび入力値を格納するRWI、出力アドレスおよび出力値を格納するRWO、依存関係格納部M、行一時格納部(一時格納部)tmp00、入力側番号格納部(入力側グループ格納部)rgpid、出力側番号格納部(出力側グループ格納部)wgpid、番号管理部(グループ管理部)busy、および、番号一時格納部(グループ一時格納部)tmp01を有している。

[0275] 依存関係格納部Mは、入力側を列指定子[I]、出力側を行指定子[O]とする2次元配列のメモリであり、各メモリ要素には0または1が記憶されるようになっている。また、依存関係格納部Mにおいて、各列はRWIに登録されている入力アドレスおよび入力値の組のそれぞれに対応しており、各行はRWOに登録されている出力アドレスおよび出力値の組のそれぞれに対応している。そして、依存関係格納部Mは、各出力アドレスおよび出力値の組が、どの入力アドレスおよび入力値の組を起源とするものであるかを示している。

[0276] 行一時格納部tmp00は、依存関係格納部Mから読み出された行を一時的に保存するメモリである。入力側番号格納部rgpidは、入力アドレスおよび入力値の組のそれぞれに対して付与するグループ番号を格納するメモリである。出力側番号格納部wgpidは、出力アドレスおよび出力値の組のそれぞれに対して付与するグループ番号を格納するメモリである。命令区間の実行完了時には、(1)各入力データが所属するグループ番号が、対応する入力側番号格納部rgpidに1が示されているビット位置によって得られ、(2)各出力データが所属するグループ番号は、対応する出力側番号格納部wgpidに1が示されているビット位置によって得られることになる。

[0277] 番号管理部busyは、グループ番号の空き状況を管理するメモリである。番号一時格納部tmp01は、入力側番号格納部rgpidおよび出力側番号格納部wgpidから読み出したグループ番号情報を一時的に格納するメモリである。

[0278] 命令区間の実行開始直前には、依存関係格納部M、行一時格納部tmp00、入力側番号格納部rgpid、出力側番号格納部wgpid、番号管理部busy、および、番号一時格納部tmp01が全て0に初期化される。そして、該命令区間のPC値がRW4Aに

おけるPCに格納される。その後、命令区間の実行が順次行われると、レジスタ／メモリからの読み出し、および／または、レジスタ／メモリへの書き込みが順に行われることになる。

[0279] 命令区間実行時にレジスタ／メモリからの読み出しが行われた場合には、RW4Aによって次の処理が行われる。

[0280] (CR1)読み出しが行われたレジスタ／メモリのアドレスが、RWOに登録されているか否かが検索される。RWOに登録されている場合には、既に出力値として入出力セットに登録されている値の読み出しが行われたものであるので、入力値として登録する必要はないことになる。すなわち、該アドレスをRWIに登録せずに終了する。

[0281] この時、RWOにおいて既に登録されているアドレスに対応する依存関係格納部Mの行成分の各メモリ要素の値が取り出され、行一時格納部tmp00に格納されている各要素との論理和が演算される。この演算結果としての論理和が、行一時格納部tmp00に格納される。すなわち、読み出しが行われたデータの起源を表す行一時格納部tmp00に、入力の情報が格納されることになる。

[0282] また、RWOにおいて既に登録されているアドレスに対応する出力側番号格納部wgpidの行成分の各要素の値が取り出され、番号一時格納部tmp01に格納されている各要素との論理和が演算される。この演算結果としての論理和が、番号一時格納部tmp01に格納される。すなわち、読み出しが行われたデータの所属グループを示す番号一時格納部tmp01に、入力の情報が格納されることになる。

[0283] (CR2)読み出しが行われたレジスタ／メモリのアドレスが、RWOに登録されていない場合には、該アドレスがRWIに登録されているか否かが検索される。RWIに登録されている場合には、既に入力値として入出力セットに登録されている値の読み出しが行われたものであるので、さらに入力値として登録する必要はないことになる。すなわち、該アドレスをRWIに登録せずに終了する。

[0284] この時、行一時格納部tmp00における、入力側の既登録位置[I]に対応するメモリ要素のビットを1とする。すなわち、読み出しが行われたデータの起源を表す行一時格納部tmp00が新規に作成されることになる。

[0285] また、行一時格納部tmp00においてビットが1にセットされたメモリ要素の列位置に

対応する入力側番号格納部rgpidが読み出され、番号一時格納部tmp01に格納されている要素との論理和が演算される。この演算結果としての論理和が、番号一時格納部tmp01に格納される。すなわち、読み出しが行われたデータの所属グループを示す番号一時格納部tmp01に、入力の情報が格納されることになる。

[0286] (CR3)読み出しが行われたレジスタ／メモリのアドレスが、RWOおよびRWIのいずれにも登録されていない場合には、該アドレスおよび値を入力アドレスおよび入力値としてRWIに登録する。

[0287] この時、行一時格納部tmp00における、新たに登録された位置[I]に対応するメモリ要素のビットを1とする。すなわち、読み出しが行われたデータの起源を表す行一時格納部tmp00が新規に作成されることになる。

[0288] また、命令区間実行時にレジスタ／メモリへの書き込みが行われた場合には、RW4Aによって次の処理が行われる。

[0289] また、行一時格納部tmp00においてビットが1にセットされたメモリ要素の列位置に対応する入力側番号格納部rgpidが読み出され、番号一時格納部tmp01に格納されている要素との論理和が演算される。この演算結果としての論理和が、番号一時格納部tmp01に格納される。すなわち、読み出しが行われたデータの所属グループを示す番号一時格納部tmp01に、入力の情報が格納されることになる。

[0290] (CW1)書き込みが行われたレジスタ／メモリのアドレスが、RWOに登録されているか否かが検索される。RWOに登録されている場合には、既に出力値として入出力セットに登録されている値の書き換えが行われたことになるので、登録されている出力アドレスに対応する出力値を、書き込みが行われた値に更新し、終了する。

[0291] この時、RWOにおいて既に登録されているアドレスに対応する依存関係格納部Mの行成分の各メモリ要素の値が、該命令区間の実行時に生成された行一時格納部tmp00に格納されている各要素の値に書き換えられる。すなわち、書き込みが行われたデータの起源が行一時格納部tmp00の値に置き換えられることになる。

[0292] ここで、番号一時格納部tmp01の各要素の値がチェックされる。そして、番号一時格納部tmp01の全ての要素が0である場合には、番号管理部busyにおいて空きグループ番号となっている列位置の1つに対応する、番号一時格納部tmp01における

列位置が1に設定される。具体的には、番号管理部busyにおける各要素のうち、最も左寄りの0に該当する列位置に対応する、番号一時格納部tmp01における列位置が1に設定される。また、この際に、番号一時格納部tmp01において、1に設定された列位置に対応する、番号管理部busyにおける列位置が1に設定される。

[0293] 一方、番号一時格納部tmp01の要素に1がある場合には、最も左寄りの1に対応する列位置が使用すべきグループ番号と認識される。そして、番号一時格納部tmp01の全ての要素のうち、最も左寄りの位置にある1を残して、残りの要素を0にした値が、出力側番号格納部wgpidにおける既登録位置[O]、および、行一時格納部tmp00の該当位置が1である入力側番号格納部rgpidの該当位置にそれぞれ書き込まれる。

[0294] さらに、番号一時格納部tmp01の全ての要素のうち、最も左寄りの位置にある1を除いた残りの値と、入力側番号格納部rgpid全体および出力側番号格納部wgpid全体とをそれぞれ比較し、各要素の論理積が求められる。そして、論理積の結果、1となる要素を保持する入力側番号格納部rgpidおよび出力側番号格納部wgpidの列位置については、該当要素を0にリセットすることにより、番号一時格納部tmp01全体のうち、最も左寄りの位置にある1を残して、残りを0にした値がセットされる。

[0295] (CW2)書き込みが行われたレジスタ／メモリのアドレスが、RWOに登録されていない場合には、該アドレスおよび値を出力アドレスおよび出力値としてRWOに登録する。

[0296] この時、新たに追加した出力アドレスおよび出力値(エントリ)に対応する依存関係格納部Mの行成分の各メモリ要素の値が、該命令区間の実行時に生成された行一時格納部tmp00に格納されている各要素の値に書き換えられる。すなわち、書き込みが行われたデータの起源が行一時格納部tmp00の値に置き換えられることになる。

[0297] ここで、上記したCW2と同様に、番号一時格納部tmp01の各要素の値がチェックされる。そして、番号一時格納部tmp01の全ての要素が0である場合には、番号管理部busyにおいて空きグループ番号となっている列位置の1つに対応する、番号一時格納部tmp01における列位置が1に設定される。具体的には、番号管理部busy

における各要素のうち、最も左寄りの0に該当する列位置に対応する、番号一時格納部tmp01における列位置が1に設定される。また、この際に、番号一時格納部tmp01において、1に設定された列位置に対応する、番号管理部busyにおける列位置が1に設定される。

[0298] 一方、番号一時格納部tmp01の要素に1がある場合には、最も左寄りの1に対応する列位置が使用すべきグループ番号と認識される。そして、番号一時格納部tmp01の全ての要素のうち、最も左寄りの位置にある1を残して、残りの要素を0にした値が、出力側番号格納部wgpidにおける既登録位置[O]、および、行一時格納部tmp00の該当位置が1である入力側番号格納部rgpidの該当位置にそれぞれ書き込まれる。

[0299] さらに、番号一時格納部tmp01の全ての要素のうち、最も左寄りの位置にある1を除いた残りの値と、入力側番号格納部rgpid全体および出力側番号格納部wgpid全体とをそれぞれ比較し、各要素の論理積が求められる。そして、論理積の結果、1となる要素を保持する入力側番号格納部rgpidおよび出力側番号格納部wgpidの列位置については、該当要素を0にリセットすることにより、番号一時格納部tmp01全体のうち、最も左寄りの位置にある1を残して、残りを0にした値がセットされる。

[0300] 以上の手順により、命令区間の実行完了時には、以下の情報が得られている。

(1)各入力データが所属するグループ番号は、対応する入力側番号格納部rgpidに1が表示されているビット位置により得られる。

(2)各出力データが所属するグループ番号は、対応する出力側番号格納部wgpidに1が表示されているビット位置により得られる。

[0301] ここで、命令区間の一例として、図15に示す命令区間を実行した場合の例について説明する。同図において、PCは、該命令区間が開始された際のPC値を示している。このPC値が、RW4APCに格納される。

[0302] 第1の命令において、メモリにおけるアドレスA1からロードした4バイトデータ(00110000)が、レジスタにおけるアドレスR1に格納される。この時点では、読み出しが行われたアドレスA1は、RWOおよびRWIのいずれにも登録されていないので、アドレスA1およびデータ(00110000)がRWIの第1列位置に登録される。

[0303] また、同時に、行一時格納部tmp00の第1列位置に1がセットされ、行一時格納部tmp00は「1000」となる。

[0304] また、行一時格納部tmp00に1がセットされた第1列位置に対応する入力側番号格納部rgpidの第1列位置から読み出された値「0000」が、番号一時格納部tmp01に書き込まれる。番号一時格納部tmp01の全ての要素が0の場合には、番号管理部busyの各要素が確認される。そして、番号管理部busy「0000」の各要素のうち、最も左寄りの0に該当する第1列位置が、次に利用すべき空きグループ番号として認識される。その後、番号管理部busyおよび番号一時格納部tmp01の第1列位置に1がそれぞれセットされる。この結果、番号管理部busyは「1000」、番号一時格納部tmp01は「1000」となる。

[0305] また、書き込みが行われたアドレスR1は、RWOに登録されていないので、アドレスR1およびデータ(00110000)がRWOの第1行に登録される。これと同時に、行一時格納部tmp00の全ての要素「1000」が依存関係格納部Mの第1行に書き込まれる。また、番号一時格納部tmp01の全ての要素「1000」が、出力側番号格納部wgpidの第1行に書き込まれる。また、同じく番号一時格納部tmp01の全ての要素「1000」が、行一時格納部tmp00において1となっている列における、入力側番号格納部rgpidの第1列位置に書き込まれる。

[0306] 次に、第2の命令において、メモリにおけるアドレスA2からロードした1バイトデータ(02)が、レジスタにおけるアドレスR2に格納される。この時点では、読み出しが行われたアドレスA2は、RWOおよびRWIのいずれにも登録されていないので、アドレスA2およびデータ(02)がRWIの第2列に登録される。この際に、アドレスA2における残り3バイトについては、Don't Careを意味する「-」が格納される。

[0307] また、同時に、行一時格納部tmp00の第2列位置に1がセットされ、行一時格納部tmp00は「0100」となる。

[0308] また、行一時格納部tmp00に1がセットされた第2列位置に対応する入力側番号格納部rgpidの第2列位置から読み出された値「0000」が、番号一時格納部tmp01に書き込まれる。番号一時格納部tmp01の全ての要素が0の場合には、番号管理部busyの各要素が確認される。そして、番号管理部busy「1000」の各要素のうち、最も

左寄りの0に該当する第2列位置が、次に利用すべき空きグループ番号として認識される。その後、番号管理部busyおよび番号一時格納部tmp01の第2列位置に1がそれぞれセットされる。この結果、番号管理部busyは「1100」、番号一時格納部tmp01は「0100」となる。

[0309] また、書き込みが行われたアドレスR2は、RWOに登録されていないので、アドレスR2およびデータ(02)がRWOの第2行に登録される。これと同時に、行一時格納部tmp00の全ての要素「0100」が依存関係格納部Mの第2行に書き込まれる。また、番号一時格納部tmp01の全ての要素「0100」が、出力側番号格納部wgpidの第2行に書き込まれる。また、同じく番号一時格納部tmp01の全ての要素「0100」が、行一時格納部tmp00において1となっている列における、入力側番号格納部rgpidの第2列位置に書き込まれる。

[0310] 次に、第3の命令において、メモリにおけるアドレス(A2+R2)からロードした1バイトデータ(22)が、レジスタにおけるアドレスR2に格納される。この場合、アドレスR2は命令区間の内部にて上書きされたレジスタであるので、アドレスR2は命令区間の入力とはならない。

[0311] 一方、アドレスR2に格納されているデータは(02)であったので、読み出しが行われたメモリにおけるアドレスは(A2+02)となる。この時点では、読み出しが行われたアドレス(A2+02)は、RWOおよびRWIのいずれにも登録されていないので、アドレス(A2+02)およびデータ(22)がRWIの第2列に追加登録される。

[0312] この際に、アドレスA2における4バイトのうち、アドレス(A2+02)となるバイトの部分にデータ(22)が登録される。すなわち、第2の命令において、アドレスA2となるバイトの部分にデータ(02)が登録されているので、アドレス(A2+01)およびアドレス(A2+03)となるバイトの部分に、Don't Careを意味する「-」が格納されたままとなる。

[0313] また、同時に、行一時格納部tmp00の第2列位置に1がセットされ、行一時格納部tmp00は「0100」となる。

[0314] また、アドレスR2からの読み出しに対応して、依存関係格納部Mのうち、アドレスR2に対応する第2行位置から読み出された値「0100」が、行一時格納部tmp00に書き込まれる。さらに、出力側番号格納部wgpidのうち、アドレスR2に対応する第2行位

置から読み出された値「0100」が番号一時格納部tmp01に書き込まれる。

[0315] また、行一時格納部tmp00に1がセットされた第2列位置に対応する入力側番号格納部rgpidの第2列位置から読み出された値「0100」が、番号一時格納部tmp01に論理和として書き込まれる。番号一時格納部tmp01にビットが1となっている要素がある場合には、最も左寄りの1に該当する第2列位置が、次に利用すべき空きグループ番号として認識される。

[0316] また、書き込みが行われたアドレスR2は、既にRWOに登録されているので、アドレスR2に対応する出力値として、データ(02)からデータ(22)に書き換えられる。これと同時に、行一時格納部tmp00の全ての要素「0100」が依存関係格納部Mの第2行に書き込まれる。また、番号一時格納部tmp01の全ての要素「0100」が、出力側番号格納部wgpidの第2行に書き込まれる。また、同じく番号一時格納部tmp01の全ての要素「0100」が、行一時格納部tmp00において1となっている列における、入力側番号格納部rgpidの第2列位置に書き込まれる。

[0317] 次に、第4の命令において、メモリにおけるアドレスA3からロードした1バイトデータ(33)が、レジスタにおけるアドレスR3に格納される。この時点では、読み出しが行われたアドレスA3は、RWOおよびRWIのいずれにも登録されていないので、アドレスA3およびデータ(33)がRWIの第3列に登録される。

[0318] また、同時に、行一時格納部tmp00の第3列位置に1がセットされ、行一時格納部tmp00は「0010」となる。

[0319] また、行一時格納部tmp00に1がセットされた第3列位置に対応する入力側番号格納部rgpidの第3列位置から読み出された値「0000」が、番号一時格納部tmp01に書き込まれる。番号一時格納部tmp01の全ての要素が0の場合には、番号管理部busyの各要素が確認される。そして、番号管理部busy「1100」の各要素のうち、最も左寄りの0に該当する第3列位置が、次に利用すべき空きグループ番号として認識される。その後、番号管理部busyおよび番号一時格納部tmp01の第3列位置に1がそれぞれセットされる。この結果、番号管理部busyは「1110」、番号一時格納部tmp01は「0010」となる。

[0320] また、書き込みが行われたアドレスR3は、RWOに登録されていないので、アドレス

R3およびデータ(33)がRWOの第3行に登録される。これと同時に、行一時格納部tmp00の全ての要素「0010」が依存関係格納部Mの第3行に書き込まれる。また、番号一時格納部tmp01の全ての要素「0010」が、出力側番号格納部wgpidの第3行に書き込まれる。また、同じく番号一時格納部tmp01の全ての要素「0010」が、行一時格納部tmp00において1となっている列における、入力側番号格納部rgpidの第3列位置に書き込まれる。

[0321] 最後に、第5の命令において、メモリにおけるアドレス(R1+R2)からロードした1バイトデータ(44)が、レジスタにおけるアドレスR4に格納される。ここで、アドレスR1およびR2は、命令区間の内部にて上書きされたレジスタであるので、命令区間の入力とはならない。一方、(R1+R2)によって生成されたアドレスA4は命令区間の入力となる。このアドレスA4は、RWOおよびRWIのいずれにも登録されていないので、アドレスA4およびデータ(44)がRWIの第4列に登録される。

[0322] また、同時に、行一時格納部tmp00の第4列位置に1がセットされ、行一時格納部tmp00は「0001」となる。

[0323] また、アドレスR1およびアドレスR2からの読み出しに対応して、依存関係格納部Mのうち、アドレスR1およびアドレスR2に対応する第1行位置および第2行位置から読み出された値「1000」、値「0100」、および行一時格納部tmp00の値「0001」の論理和「1101」が、行一時格納部tmp00に書き込まれる。さらに、出力側番号格納部wgpidのうち、アドレスR1およびアドレスR2に対応する第1行位置および第2行位置から読み出された値「1000」および値「0100」の論理和「1100」が番号一時格納部tmp01に書き込まれる。

[0324] また、最初に行一時格納部tmp00に1がセットされた第4列位置に対応する入力側番号格納部rgpidの第4列位置から読み出された値「0000」と、番号一時格納部tmp01に格納されている値「1100」との論理和「1100」が、番号一時格納部tmp01に書き込まれる。番号一時格納部tmp01にビットが1となっている要素がある場合には、最も左寄りの1に該当する第1列位置が、次に利用すべき空きグループ番号として認識される。

[0325] また、書き込みが行われたアドレスR4は、RWOに登録されていないので、アドレス

R4およびデータ(44)がRWOの第4行に登録される。これと同時に、行一時格納部tmp00の全ての要素「1101」が依存関係格納部Mの第4行に書き込まれる。また、番号一時格納部tmp01の全ての要素「1100」のうち、最も左寄りの1を残して、残りを0にした「1000」が、出力側番号格納部wgpidの第4行に書き込まれる。また、同じく番号一時格納部tmp01の全ての要素「1100」のうち、最も左寄りの1を残して、残りを0にした「1000」が、行一時格納部tmp00において1となっている列における、入力側番号格納部rgpidの第1、2、4列位置に書き込まれる。

[0326] さらに、番号一時格納部tmp01の全ての要素「1100」のうち、最も左寄りの1を除外した残りの「0100」を、入力側番号格納部rgpidの全ての要素および出力側番号格納部wgpidの全ての要素と比較し、論理積が1になるビットを保持する入力側番号格納部rgpidおよび出力側番号格納部wgpidのエントリについては、該当ビットが0にリセットされることにより、番号一時格納部tmp01全体「1100」のうち、最も左寄りの位置にある1を残して、残りを0にした値「0100」がセットされる。

[0327] 以上の手順により、入力セットA1—D1、A2—D2、およびA4—D4については、対応する入力側番号格納部rgpidの列位置にグループ0が表示され、出力セットR1、R2、およびR4については、対応する出力側番号格納部wgpidの行位置にグループ0が表示される。一方、入力セットA3—D3については、対応する入力側番号格納部rgpidの列位置にグループ番号2が表示され、出力セットR3については、対応する出力側番号格納部wgpidの行位置にグループ2が表示される。

[0328] (RWの第4構成例)

ある命令区間に条件分岐命令が含まれている場合、条件分岐に使用した条件コードの生成に関与した資源が、条件分岐命令実行後の全ての命令の実行に関与することになる。よって、条件分岐命令を考慮したグループ分割技術も必要となる。以下に示すRWの第4構成例は、これを実現することが可能となっている。

[0329] 図21は、第4構成例としてのRW4Aの概略構成を示している。同図に示すように、RW4Aは、命令区間の命令区間のPC値を格納するPC、入力アドレスおよび入力値を格納するRWI、出力アドレスおよび出力値を格納するRWO、依存関係格納部M、行一時格納部tmp00、入力側番号格納部rgpid、出力側番号格納部wgpid、番号

管理部busy、および、番号一時格納部tmp01に加えて、条件分岐命令の実行に伴う依存関係の擾乱に追随することを目的として、条件分岐格納部(条件分岐格納部)tmpccを有している。

[0330] 条件分岐格納部tmpccは、条件分岐に関わった入力セットを格納するメモリである。条件分岐の成立／不成立に拘らず、分岐命令後の命令を実行すること自体が条件コード生成に関わった全入力に依存することになる。

[0331] 前記したRWの第3構成例では、各命令ごとに行一時格納部tmp00を初期化するのに対して、RWの第4構成例では、条件分岐格納部tmpccの値を行一時格納部tmp00にコピーすることをもって、行一時格納部tmp00の初期化としている。

[0332] 図21に示す例では、図22に示す命令区間が実行された場合を示している。図22に示す命令区間において、第1の命令から第4の命令までは、図15に示した命令区間と同様である。

[0333] 第5の命令において、アドレスR3の値が検査される。そして、第6の命令において、第5の命令による検査結果に基づいて条件分岐が行われる。ここで、第5および第6の命令は、アドレスR3の値に依存しているので、実行時には、依存関係格納部Mの第3行が条件分岐格納部tmpccにコピーされる。そして、第7の命令が実行される前に、条件分岐格納部tmpccに格納されている「0010」が行一時格納部tmp00にコピーされる。

[0334] 第7の命令では、アドレス(R1+R2)からロードした1バイトデータ(44)が、レジスタにおけるアドレスR4に格納される。ここで、アドレスR1およびR2は、命令区間の内部にて上書きされたレジスタであるので、命令区間の入力とはならない。一方、(R1+R2)によって生成されたアドレスA4は命令区間の入力となる。このアドレスA4は、RWOおよびRWIのいずれにも登録されていないので、アドレスA4およびデータ(44)がRWIの第4列に登録される。

[0335] また、同時に、行一時格納部tmp00の第4列位置に1がセットされ、行一時格納部tmp00は「0011」となる。

[0336] また、アドレスR1およびアドレスR2からの読み出しに対応して、依存関係格納部Mのうち、アドレスR1およびアドレスR2に対応する第1行位置および第2行位置から読

み出された値「1000」、値「0100」、および行一時格納部tmp00の値「0011」の論理和「1111」が、行一時格納部tmp00に書き込まれる。さらに、出力側番号格納部wgpidのうち、アドレスR1およびアドレスR2に対応する第1行位置および第2行位置から読み出された値「1000」および値「0100」の論理和「1100」が番号一時格納部tmp01に書き込まれる。

[0337] また、最初に行一時格納部tmp00に1がセットされた第3列位置および第4列位置に対応する入力側番号格納部rgpidの第3列位置および第4列位置から読み出された値「0010」および値「0000」の論理和「0010」と、番号一時格納部tmp01に格納されている値「1100」との論理和「1110」が、番号一時格納部tmp01に書き込まれる。番号一時格納部tmp01にビットが1となっている要素がある場合には、最も左寄りの1に該当する第1列位置が、次に利用すべき空きグループ番号として認識される。

[0338] また、書き込みが行われたアドレスR4は、RWOに登録されていないので、アドレスR4およびデータ(44)がRWOの第4行に登録される。これと同時に、行一時格納部tmp00の全ての要素「1111」が依存関係格納部Mの第4行に書き込まれる。また、番号一時格納部tmp01の全ての要素「1110」のうち、最も左寄りの1を残して、残りを0にした「1000」が、出力側番号格納部wgpidの第4行に書き込まれる。また、同じく番号一時格納部tmp01の全ての要素「1110」のうち、最も左寄りの1を残して、残りを0にした「1000」が、行一時格納部tmp00において1となっている列における、入力側番号格納部rgpidの第1、2、3、4列位置に書き込まれる。

[0339] さらに、番号一時格納部tmp01の全ての要素「1110」のうち、最も左寄りの1を除外した残りの「0110」を、入力側番号格納部rgpidの全ての要素および出力側番号格納部wgpidの全ての要素と比較し、論理積が1になるビットを保持する入力側番号格納部rgpidおよび出力側番号格納部wgpidのエントリについては、該当ビットが0にリセットされることにより、番号一時格納部tmp01全体「1110」のうち、最も左寄りの位置にある1を残して、残りを0にした値「1000」がセットされる。

[0340] 以上の手順により、入力セットA1—D1、A2—D2、A3—D3、およびA4—D4については、対応する入力側番号格納部rgpidの列位置にグループ0が表示され、出力セットR1、R2、R3、およびR4については、対応する出力側番号格納部wgpidの行位

置にグループ0が表示される。すなわち、条件分岐命令の実行によって、アドレスR4の出力が依存する入力データは、入力セットA1−D1、A2−D2、A3−D3、およびA4−D4の全てとなっている。

[0341] (グループ分割に基づく再利用表設定)

以上のように、RWの第3および第4構成例によれば、各入力セットおよび各出力セットに対して、所属するグループ番号が割り当てられることになる。これに基づいて、同一グループとなる入力セットごとに、図19に示すような独立した木構造を構成することが可能となる。この木構造に基づいて、図23に示すように、各入力セットがRBおよびRFの連想検索装置に格納される。例えば図3に示す例では、初期検索キーとしてFFのみが用いられているが、上記のように各入力セットにグループ番号が与えられることによって、複数の初期検索キー(F0、F1、F2、…)を用いることが可能となる。これにより、複数の検索を同時に開始することが可能となる。図23に示す例では、グループ番号が0および2となっている独立木構造の先頭キーに対して、初期検索キーF0およびF2が設定されている。

[0342] (レジスタ値の詳細)

レジスタ入出力値としては、引数、返り値(Args.)、および、引数および返り値以外のレジスタおよび条件コード(Regs.,CC)が挙げられる。本実施形態では、SPARCアーキテクチャレジスタのうち、汎用レジスタ%g0−7、%o0−7、%l0−7、%i0−7、浮動小数点レジスタ%f0−31、条件コードレジスタICC、浮動小数点条件コードレジスタFCCを用いるようになっている(詳細は後述する)。このうち、リーフ関数の入力は汎用レジスタ%o0−5、出力は汎用レジスタ%o0−1または%f0−1、また、非リーフ関数の入力は汎用レジスタ%i0−5、出力は汎用レジスタ%i0−1または%f0−1、になり、入力は、arg[0−5]、出力は、rti[0−1] または%rtf[0−1]に登録される。SPARC−ABIの規定では、これら以外のレジスタは関数の入出力にはならないので、関数に関しては、レジスタ入出力値としては、Args.がRB、およびRO1／RO2に登録されることになる。

[0343] 一方、SPARC−ABIの規定では、ループの入出力に関しては、用いられるレジスタの種類を特定することはできないので、ループの入出力を特定するには、全ての種類のレジスタに関してRBに登録する必要がある。よって、ループに関しては、レジス

タ入出力値として、Regs.,CCに相当する、%g0-7、%o0-7、%l0-7、%i0-7、%f0-31、ICC、FCCが登録されることになる。

[0344] (多重再利用)

1レベルで上記のような再利用機構を用いた場合、図46(a)に示した例で言えば、リーフ関数としての関数Bや、関数Bの内部にあるループCなどをそれぞれ再利用することが可能となる。これに対して、ある関数を一度実行しただけで、その関数の内部に含まれる関数やループを含む全ての命令区間が再利用可能となるように登録を行う仕組みが多重再利用である。例えば上記の例で言えば、多重再利用によれば、関数Aを一度実行しただけで、入れ子関係にあるA, B, Cの全ての命令区間が再利用可能となる。以下に、多重再利用を実現する上で必要とされる機能拡張について説明する。

[0345] 図6に、一例として、関数Aおよび関数Dの概念的な構造を示す。同図に示す例では、関数Aの内部にループBが存在しており、ループBの内部にループCが存在しており、ループCにおいて関数Dが呼び出されるようになっている。そして、関数Dの内部にループEが存在しており、ループEの内部にループFが存在している。

[0346] 図7は、図6に示す関数A, DおよびループB, C, E, Fの入れ子構造において、内側の構造のレジスタ入出力(太枠セル領域)が、外側の構造のレジスタ入出力となる影響範囲(矢印)について示している。例えば、ループFの内部において入力として参照された%i0ー5は、ループEおよび関数Dに対する入力でもあり、さらに、関数Dを呼び出したループCおよびループBに対する入力(ただし%o0ー5に読み替える)でもある。一方、関数Aにとって%o0ー5は局所変数に相当するので、%i0ー5(%o0ー5)は、関数Aに対してのレジスタ入力とはならない。すなわち、%i0ー5(%o0ー5)の影響範囲はループBまでとなる。別の見方をすれば、関数Dの内部で%i0ー5が参照された場合には、ループBが直接的に%o0ー5を参照しなくても、%o0ー5をループBの入力値として登録する必要がある。ループF内部において出力された%i0ー1についても同様である。

[0347] 浮動小数点レジスタはレジスタウィンドウに含まれないので、出力された%f0ー1は、関数Aを含む全階層の出力となる。一方、その他のレジスタ入出力は、関数を超え

て影響がおよぶことはない。すなわち、ループF内部における入出力、すなわち、レジスタ入力としての%i6ー7、%g,l,o、%f0ー31、%icc、%fcc、およびレジスタ出力としての%l2ー7、%g,l,o、%f2ー31、%icc、%fccの影響範囲はループEまでとなる。主記憶に対する入出力については、前述した、関数呼び出し直前の%sp(SP)と比較する方法を入れ子の全階層に対して適用することにより、影響範囲を特定することができる。

[0348] ここで、上記のようなRW4A、RW4B、および命令区間記憶部2の構成によれば、複数の命令区間の入出力を個別に記録することが可能であるので、多重再利用を実現することが可能となる。

[0349] (並列事前実行)

以上に述べた、関数やループの多重再利用では、同一パラメータが出現する間隔が長い場合や、パラメータが単調に変化し続ける場合には全く効果がないことになる。すなわち、RBエントリの生存時間よりも同一パラメータが出現する間隔が長い場合には、ある関数またはループがRBに登録されたとしても、その登録された関数またはループに関して同一パラメータが次に出現した際には、すでにその関数またはループがRBエントリから消えていることになり、再利用できないことになる。また、パラメータが単調に変化し続ける場合には、該当する関数やループがRBに登録されていても、パラメータが異なることによって再利用できないことになる。

[0350] これに対して、多重再利用を行うプロセッサとしてのMSP1Aとは別に、命令区間の事前実行によってRBエントリを有効にするプロセッサとしてのSSP1Bを複数個設けることによって、さらなる高速化を図ることができる。

[0351] 並列事前実行機構を行うためのハードウェア構成は、前記した図2に示すような構成となる。同図に示すように、RW4A・4B、演算器5A・5B、レジスタ6A・6B、キャッシュ7A・7Bは、各プロセッサごとに独立して設けられている一方、命令区間記憶部2、および主記憶3は全てのプロセッサが共有するようになっている。同図において、破線は、MSP1AおよびSSP1Bが命令区間記憶部2に対して入出力を登録するパスを示している。

[0352] ここで、並列事前実行を実現する上での課題は、(1)どのように主記憶一貫性を保

つか、(2)どのように入力を予測するか、の2点が挙げられる。以下に、これらの課題に対する解決手法について説明する。

[0353] (主記憶一貫性に関する課題の解決方法)

まず、上記の課題(1)どのように主記憶一貫性を保つかについて説明する。特に予測した入力パラメータに基づいて命令区間を実行する場合、主記憶に書き込む値がMSP1AとSSP1Bとで異なることになる。これを解決するために、図2に示すように、SSP1Bは、RBへの登録対象となる主記憶参照には命令区間記憶部2、また、その他の局所的な参照にはSSP1Bごとに設けた局所メモリとしてのLocal7Bを使用することとし、Cache7Bおよび主記憶3への書き込みを不要としている。なお、MSP1Aが主記憶3に対して書き込みを行った場合には、対応するSSP1Bのキャッシュラインが無効化される。

[0354] 具体的には、命令区間記憶部2への登録対象のうち、読み出しが先行するアドレスについては主記憶3を参照し、MSP1Aと同様にアドレスおよび値をRBへ登録する。以後、主記憶3ではなく命令区間記憶部2を参照することによって、他のプロセッサからの上書きによる矛盾の発生を避けることができる。局所的な参照については、読み出しが先行するということは、変数を初期化せずに使うことに相当し、値は不定でよいことになるので、主記憶3を参照する必要はない。

[0355] なお、局所メモリとしてのLocal7Bの容量は有限であり、関数フレームの大きさがLocal7Bの容量を超えた場合など、実行を継続できない場合は、事前実行を打ち切るようにする。また、事前実行の結果は主記憶3に書き込まれないので、事前実行結果を使って、さらに次の事前実行を行うことはできない。

[0356] (入力の予測方法)

次に、上記の課題(2)どのように入力を予測するかについて説明する。事前実行に際しては、命令区間記憶部2の使用履歴に基づいて将来の入力を予測し、SSP1Bへ渡す必要がある。このために、命令区間記憶部2に記憶されている各入力パターンごとに小さなプロセッサを設け、MSP1AやSSP1Bとは独立して入力予測値を求めるようにする。

[0357] 具体的には、最後に出現した引数(B)および最近出現した2組の引数の差分(D)

に基づいて、ストライド予測を行う。なお、B＋Dに基づく命令区間の実行はMSP1Aがすでに開始していると考える。SSP1BがN台の場合には、用意する入力予測値は、B＋D×2からB＋D×（N＋1）までの範囲とする。

[0358] 以上のように入力予測を行えば、上記した入力パラメータが単調に変化し続けるような場合に、事前に予測しておいた結果に基づいて効果的に再利用を行うことが可能となる。

[0359] ＜実施の形態2＞

本発明の他の実施形態について図面に基づいて説明すると以下の通りである。

[0360] （データ処理装置の構成）

本実施形態に係るデータ処理装置の概略構成を図25に示す。同図に示すように、該データ処理装置は、MSP1A、SSP1B、再利用表としてのRF／RB（命令列記憶手段）2'、および主記憶（主記憶手段）を備えた構成となっており、主記憶3に記憶されているプログラムデータなどを読み出して各種演算処理を行い、演算結果を主記憶3に書き込む処理を行うものである。なお、同図に示す構成では、SSP1Bを1つ備えた構成となっているが、2つ以上備えた構成となっていてもよい。

[0361] RF／RB2'は、プログラムにおける関数およびループを再利用するためのデータを格納するメモリ手段であり、RB登録処理部（登録処理手段）2Aおよび予測処理部（予測処理手段）2Bを備えた構成となっている。このRF／RB2'の詳細、ならびにRB登録処理部2Aおよび予測処理部2Bの詳細については後述する。

[0362] 主記憶3は、MSP1AおよびSSP1Bの作業領域としてのメモリであり、例えばRAMなどによって構成されるものである。例えばハードディスクなどの外部記憶手段からプログラムやデータなどが主記憶3に読み出され、MSP1AおよびSSP1Bは、主記憶3に読み出されたデータに基づいて演算を行うことになる。

[0363] MSP1Aは、RW（再利用記憶手段）4A、演算器（第1の演算手段）5A、レジスタ6A、およびCache7Aを備えた構成となっている。また、SSP1Bは、同様に、RW（再利用記憶手段）4B、演算器（第2の演算手段）5B、レジスタ6B、およびCache／Local7Bを備えた構成となっている。

[0364] RW4A・4Bは、再利用ウィンドウであり、現在実行中かつ登録中であるRFおよびR

B(後述する)の各エントリをリング構造のスタックとして保持するものである。このRW4A・4Bは、実際のハードウェア構造としては、RF／RB2'における特定のエントリをアクティブにする制御線の集合によって構成される。

[0365] 演算器5A・5Bは、レジスタ6A・6Bに保持されているデータに基づいて演算処理を行うものであり、ALUと呼ばれるものである。レジスタ6A・6Bは、演算器5A・5Bによって演算を行うためのデータを保持する記憶手段である。なお、本実施形態では、演算器5A・5B、およびレジスタ6A・6Bは、SPARCアーキテクチャに準じたものとする。Cache7A・7Bは、主記憶3と、MSP1AおよびSSP1Bとの間でのキャッシュメモリとして機能するものである。なお、SSP1Bでは、Cache7Bには、局所メモリとしてのLocal7Bが含まれているものとする。

[0366] (RF／RBの構成)

図24は、本実施形態におけるRF／RB2'によって実現される再利用表を示している。同図に示すように、RFは、複数のエントリを格納しており、各エントリに対して、該エントリが有効であるか否かを示すV、エントリ入れ替えのヒントを示すLRU、関数の先頭アドレスを示すStart、参照すべき主記憶アドレスを示すRead/Write、および、関数とループとを区別するF/Lを保持している。

[0367] また、RBは、RFに格納されているエントリに対応して複数のエントリを格納しており、各エントリに対して、該エントリが有効であるか否かを示すV、エントリ入れ替えのヒントを示すLRU、関数またはループを呼び出す際の直前のスタックポイント%spを示すSP、引数(Args.)(V:有効エントリ、Val.:値)、主記憶値(C-FLAG:Readアドレスの変更フラグ、P-Mask:Readアドレスの履歴マスク、Mask:Read/Writeアドレスの有効バイト、Value:値)、返り値(Return Values)(V:有効エントリ、Val.:値)、ループの終了アドレス(End)、ループ終了時の分岐方向を示すtaken/not、および、引数や返り値以外のレジスタおよび条件コード(Regs.,CC)を保持している。また、RBは、1つ以上のレジスタアドレスに対応して定数フラグ(Const-FLAG)を格納するメモリ領域を保持している。なお、定数フラグ(Const-FLAG)の詳細については後述する。

[0368] 上記のRFおよびRBにおける各項目についてより詳細に説明する。上記Vは、上記のようにエントリが有効であるか否かを示すものであるが、具体的には、未登録時に

は「0」、登録中である場合には「2」、登録済である場合には「1」の値が格納されるようになっている。例えば、RFまたはRBを確保する際に、未登録エントリ(V=0)があれば、これを使用し、未登録エントリがなければ、登録済エントリ(V=1)の中からLRUが最小のものを選択して上書きすることになる。登録中エントリ(V=2)は使用中であるので上書きすることはできない。

[0369] 上記LRUは、一定時間間隔で右へシフトされていくシフトレジスタの中の「1」の個数を示したものである。RFの場合、このシフトレジスタは、該当エントリに関して、再利用のための登録を行ったか、もしくは再利用を試みた場合に、左端に「1」が書き込まれるようになっている。したがって、該当エントリが頻繁に使用されれば、LRUは大きな値となり、一定期間使用されなければ、LRUの値は0となる。一方、RBの場合、シフトレジスタには、該当エントリが再利用された場合に「1」が書き込まれるようになっている。したがって、該当エントリが頻繁に使用されれば、LRUは大きな値となり、一定期間使用されなければ、LRUの値は0となる。

[0370] 上記RBにおける主記憶値のMaskについて説明する。一般に、アドレスとデータとを1バイトずつ管理することにすれば管理が可能であるが、実際には、4バイト単位でデータを管理する方がキャッシュ参照を高速に行うことができる。そこで、RFでは、主記憶アドレスを4の倍数で記憶するようになっている。一方、管理単位を4バイトとする場合、1バイト分だけをロードすることに対応できるようにするために、4バイトのうちでどのバイトが有効であるかを示す必要がある。すなわち、Maskは、4バイトのうちでどのバイトが有効であるかを示す4ビットのデータとなっている。例えば、C001番地から1バイト分をロードした結果、値がE8であった場合、RFには、アドレスC000が登録され、RBのMaskに「0100」、Valueに「00E80000」が登録されることになる。なお、Readアドレスにおける変更フラグ(C-FLAG)および履歴マスク(P-Mask)の詳細については後述する。

[0371] 上記の引数や返り値以外のレジスタおよび条件コード(Regs.,CC)について説明する。本実施形態では、SPARCアーキテクチャレジスタのうち、汎用レジスタ%g0-7、%o0-7、%l0-7、%i0-7、浮動小数点レジスタ%f0-31、条件コードレジスタICC、浮動小数点条件コードレジスタFCCを用いるようになっている(詳細は後述する)。このうち、リ

ーフ関数の入力は汎用レジスタ%o0-5、出力は汎用レジスタ%o0-1、また、非リーフ関数の入力は汎用レジスタ%i0-5、出力は汎用レジスタ%i0-1、になり、入力は、arg[0-5]、出力は、rti[0-1]に登録される。SPARC-ABIの規定では、これら以外のレジスタは関数の入出力にはならないので、関数に関してはRBにおける引数(Args.)の項で十分である。

[0372] 一方、SPARC-ABIの規定では、ループの入出力に関しては、用いられるレジスタの種類を特定することはできないので、ループの入出力を特定するには、全ての種類のレジスタに関してRBに登録する必要がある。よって、RBにおけるRegs.,CCには、%g0-7、%o0-7、%l0-7、%i0-7、%f0-31、ICC、FCCが登録されるようになっている。

[0373] 以上のように、RF／RB2'において、Readアドレスは RFが一括管理し、MaskおよびValueはRBが管理している。これにより、Readアドレスの内容とRBの複数エントリをCAMによって一度に比較する構成を可能としている。

[0374] (再利用処理の概略)

次に、関数およびループのそれぞれの場合について、再利用処理の概略について説明する。

[0375] まず、関数の場合について説明する。関数から復帰するまでに次の関数を呼び出した場合、または、登録すべき入出力が再利用表の容量を超える、引数の第7ワードを検出する、途中でシステムコールや割り込みが発生する、などの擾乱が発生しなかった場合、復帰命令を実行した時点で、登録中の入出力表エントリを有効にする。

[0376] 以降、図24を参照しながら説明すると、関数を呼び出す前に、(1)RFに登録されているエントリにおける関数の先頭アドレスに、該当関数の先頭アドレスと一致するものがあるかを検索する。一致するものがある場合には、(2)RBに登録されている該当関数に関するエントリにおける引数が、呼び出す関数の引数と完全に一致するエントリを選択する。そして、(3)関連する主記憶アドレスすなわち少なくとも1つのMaskが有効であるReadアドレスをRFからすべて参照して、(4)RBに登録されている内容と一致比較を行う。全ての入力が一致した場合に、(5)RBに登録済の出力(返り値、大域変数、およびAの局所変数)を主記憶3に書き戻すことによって、関数の実行を省略する、すなわち関数の再利用を実現することができる。

[0377] 次に、ループの場合について説明する。ループが完了する以前に関数から復帰したり、前記した擾乱が発生したりするなど、ループの入出力登録が中止されなければ、登録中のループに対応する後方分岐命令を検出した時点で、登録中の入出力表エントリを有効にし、そのループの登録を完了する。

[0378] さらに、後方分岐命令が成立する場合は、次のループが再利用可能かどうかを判断する。すなわち、図24を参照しながら説明すると、後方分岐する前に、(1)RFに登録されているエントリにおけるループの先頭アドレスに、該当ループの先頭アドレスと一致するものがあるかを検索する。一致するものがある場合には、(2)RBに登録されている該当ループに関するレジスタ入力値が、呼び出すループのレジスタ入力値と完全に一致するエントリを選択する。そして、(3)関連する主記憶アドレスをRFから全て参照して、(4)RBに登録されている内容と一致比較を行う。全ての入力が一致した場合に、(5)RBに登録済の出力(レジスタおよび主記憶出力値)を主記憶3に書き戻すことによってループの実行を省略する、すなわちループの再利用を実現することができる。

[0379] 再利用した場合、RBに登録されている分岐方向に基づいて、さらに次のループに関して同様の処理を繰り返す。一方、次のループが再利用不可能であれば、次のループを通常に実行し、RFおよびRBへの登録を開始する。

[0380] (命令区間の実行時における処理の流れ)

次に、命令がデコードされた場合の具体的な処理の流れについて説明する。以下では、命令がデコードされた結果、関数呼び出し命令である場合、関数復帰命令である場合、後方分岐成立の場合、後方分岐不成立の場合、およびその他の命令の場合について、それぞれ処理の流れを説明する。

[0381] (関数呼び出し命令である場合)

命令がデコードされた結果、関数呼び出し命令である場合の処理を図26に示すフローチャートを参照しながら以下に説明する。まずステップ1(以降、S1のように称する)において、引数の第7ワードを検出したか否かが判定される。S1においてYES、すなわち、引数の第7ワードを検出したと判定された場合には、RWに登録されている登録中RBエントリを全て無効化し、S6に移行して、プログラムカウンタを関数の先

頭へ進め、処理を終了する。

[0382] 一方、S1においてNO、すなわち、引数の第7ワードを検出していないと判定された場合には、該関数呼び出しおよび入力値がRFおよびRBに登録されているか否かを検索する(S2)。S2においてYES、すなわち、該関数呼び出しおよび入力値がRFおよびRBに登録されていると判定された場合には、後述するS7のステップに移行する。

[0383] S2においてNO、すなわち、該関数呼び出しおよび入力値がRFおよびRBに登録されていないと判定された場合、該関数のためのRFエントリおよびRBエントリを確保しようと試み、(1)既存のRFエントリがあるか、(2)登録作業中につき追い出すことのできないRFエントリ以外に、使用可能なRFエントリがあるか、または(3)登録作業中につき追い出すことができないRBエントリ以外に、使用可能なRBエントリがあるかを判定する(S3)。

[0384] S3においてNO、すなわち、使用可能なRF・RBエントリがないと判定された場合には、登録を開始せず、RWに登録されているRBを全て無効化し(S5)、RWを空にする。一方、S3においてYES、すなわち、使用可能なRF・RBエントリがあると判定された場合には、該関数のためのRFエントリおよびRBエントリを確保し、RWに登録する(S4)。ここで、RWに登録した際に、RWに登録されているRWエントリが溢れた際には、最も古いRWエントリを削除し、対応するRBを無効化する。S3またはS4が行われた後に、プログラムカウンタを関数の先頭へ進め(S6)、処理を終了する。

[0385] 一方、S2においてYES、すなわち、該関数呼び出しおよび入力値がRFおよびRBに登録されていると判定された場合、該関数は再利用可能であることになる。すなわち、RBから出力値を求めるとともに、レジスタおよび主記憶3にこの出力値を書き込む(S7)。そして、登録中の関数／ループがRWに登録されているか否かを判定し(S8)、登録されている場合には、再利用を行った関数のRBエントリの内容のうち必要なものをRWに登録されているエントリに追加する(S9)。ここで、RWのTOPから順に登録し、途中でRBがあふれた場合には、以降、RWのBOTTOMまでに対するRBを無効化し、RWから削除する。その後、プログラムカウンタを次の命令へ進め(S10)、処理を終了する。

[0386] （関数復帰命令である場合）

命令がデコードされた結果、関数復帰命令である場合の処理を図27に示すフローチャートを参照しながら以下に説明する。S11において、RWのTOPから順にたどり、関数に対応するRF／RBが検出されるまでに、ループに関するRBが検出されるか否かが判定される（S12）。ここでループに関するRBが検出されると（S12においてYES）、該当RBを全て無効化するとともに、RWから削除する（S13）。

[0387] 一方、RW探索中に、該関数に対応するRF／RBを検出したか否かが判定される（S14）。ここで、該関数に対応するRF／RBが検出されると（S14においてYES）、該当RBエントリを有効化するとともに、RWから削除する（S15）。

[0388] その後、復帰命令を実行し（S16）、処理を終了する。

[0389] （後方分岐成立である場合）

命令がデコードされた結果、後方分岐成立である場合の処理を図28に示すフローチャートを参照しながら以下に説明する。まず、RWのTOPから順にたどり、関数に対応するRBを検出するか否かが判定される（S21）。S21においてYES、すなわち、関数に対応するRBを検出した場合には、後述するS24のステップに移行する。

[0390] 一方、S21においてNO、すなわち、関数に対応するRBを検出しない場合には、次に、該後方分岐命令自身のアドレスとRB中のループ終了アドレスとが一致するか否かが判定される（S22）。S22においてNO、すなわち、該後方分岐命令自身のアドレスとRB中のループ終了アドレスとが一致しないと判定されると、後述するS24のステップに移行する。

[0391] S22においてYES、すなわち、該後方分岐命令自身のアドレスとRB中のループ終了アドレスとが一致すると判定された場合、RWのTOPから該RBの手前までのRBを全て無効化し（S23）、RWから削除する。また、該RBエントリを有効化し、かつtaken=1とし、RWから削除する。

[0392] 次に、S24において、次ループの先頭アドレスおよび入力値がRFおよびRBに登録されているか否かが判定される。S24においてYES、すなわち、次ループの先頭アドレスおよび入力値がRFおよびRBに登録されている場合には、後述するS30のステップに移行する。

[0393] 一方、S24においてNO、すなわち、次ループの先頭アドレスおよび入力値がRFおよびRBに登録されていない場合には、次ループのためのRFエントリおよびRBエントリを確保しようと試み、(1)既存のRFエントリがあるか、(2)登録作業中につき追い出すことができないRFエントリ以外に、使用可能なRFエントリがあるか、または(3)登録作業中につき追い出すことができないRBエントリ以外に、使用可能なRBエントリがあるかが判定される(S25)。

[0394] S25においてNO、すなわち、使用可能なRF・RBエントリがないと判定された場合には、登録を開始せずに、RWに登録されているRBを全て無効化し(S26)、RWを空にする。その後、S29において、プログラムカウンタを条件分岐先へ進め、処理を終了する。

[0395] 一方、S25においてYES、すなわち、使用可能なRF・RBエントリがあると判定された場合には、その使用可能なRF・RBエントリを確保し、確保したRF・RBをRWに登録する(S27)。また、RBにループ終了アドレス(後方分岐命令自身のアドレス)を登録する。ここで、RWへの登録を行った際にRWが溢れた場合には、最も古いRWエントリを削除し(S28)、それに対応するRBを無効化する。その後、S29において、プログラムカウンタを条件分岐先へ進め、処理を終了する。

[0396] 一方、前記したS24においてYESとなった場合、次ループは再利用可能であることになるので、RBから出力値を求め、この値をレジスタおよび主記憶3に書き込む(S30)。ここで、登録中の関数／ループがRWに登録されているか否かが判定され(S31)、登録されている場合、再利用を行ったループのRBエントリの内容のうち必要なものをRWに登録されているエントリに追加する(S32)。このとき、RWのTOPから順に登録し、途中でRBが溢れた場合、以降、RWのBOTTOMまでに対するRBを無効化し、RWから削除する。

[0397] その後、プログラムカウンタは、次ループ先頭ではなく、該RB中のtakenの値に応じて、taken=1の場合は自命令、taken=0の場合は、RB中に記憶しておいたループ終了アドレスの次へ進める。その後、処理を終了する。

[0398] (後方分岐不成立である場合)

命令がデコードされた結果、後方分岐不成立である場合の処理を図29に示すフロ

ーチャートを参照しながら以下に説明する。まず、RWのTOPから順に検索し(S41)、関数に対応するRBを検出したか否かが判定される(S42)。S42においてYES、すなわち、関数に対応するRBを検出したと判定された場合、S46においてプログラムカウンタを次命令に進め、処理を終了する。

[0399] S42においてNO、すなわち、関数に対応するRBを検出していないと判定された場合、該後方分岐命令自身のアドレスとRB中のループ終了アドレスが一致するか否かが判定される(S43)。S43においてNO、すなわち、該後方分岐命令に対応するRF／RBを検出していないと判定された場合、S46においてプログラムカウンタを次命令に進め、処理を終了する。

[0400] 一方、S43においてYES、すなわち、該後方分岐命令に対応するRF／RBを検出したと判定された場合、RWのTOPから該RBの手前までのRBを全て無効化し(S44)、RWから削除する。また、該RBエントリを有効化し、かつtaken=0とし、RWから削除する(S45)。その後、S46においてプログラムカウンタを次命令に進め、処理を終了する。

[0401] (その他の命令である場合)

次に、命令がデコードされた結果、上記以外のその他の命令である場合について説明する。その他の命令である場合、レジスタR／W、主記憶R／Wが実行される。その際に、RWが空でなければ、以下の手順によってレジスタR／W、主記憶R／WをRWに登録されているRBに対して登録する。以下では、(1)汎用レジスタREADの場合、(2)汎用レジスタWRITEの場合、(3)浮動小数点レジスタREADの場合、(4)浮動小数点レジスタWRITEの場合、(5)条件コードレジスタICC—READの場合、(6)条件コードレジスタICC—WRITEの場合、(7)浮動小数点条件コードレジスタFCC—READの場合、(8)浮動小数点条件コードレジスタFCC—WRITEの場合、(9)主記憶READの場合、(10)主記憶WRITEの場合についてそれぞれ説明する。

[0402] (1)汎用レジスタREADの場合

まず、RWのTOPからBOTTOMまで順にたどる。そして、(1—1)該RBがリーフ関数かつ%o0-6の場合、または該RBが非リーフ関数かつ%i0-6の場合、arg[0-5].V=0であれば、arg[0-5].V=1に変更し、arg[0-5].Valに読み出しデータを記録する。その後、

さらにRWをたどり、該RBが関数の場合、処理を終了する。一方、該RBが関数ではない(ループである)場合、arg[0-5].V=0であれば、arg[0-5].V=1に変更し、arg[0-5].Valに読み出しデータを記録し、処理を終了する。

[0403] 一方、(1-2)該RBがループの場合、(a)%g0-7でgrr[0-7].V=0であれば、grr[0-7].V=1に変更し、grr[0-7].Valに読み出しデータを記録し、処理を終了する。(b)%o0-7でarg[0-7].V=0であれば、arg[0-7].V=1に変更し、arg[0-7].Valに読み出しデータを記録し、処理を終了する。(c)%l0-7でlrr[0-7].V=0であれば、lrr[0-7].V=1に変更し、lrr[0-7].Valに読み出しデータを記録し、処理を終了する。(d)%i0-7でirr[0-7].V=0であれば、irr[0-7].V=1に変更し、irr[0-7].Valに読み出しデータを記録し、次のRWエントリに進む。

[0404] (2)汎用レジスタWRITEの場合

まず、RWのTOPからBOTTOMまで順にたどる。そして(2-1)該RBがリーフ関数かつ%o0-5の場合、または該RBが非リーフ関数かつ%i0-5の場合、arg[0-5].V=0であれば、以降の読み出しは入力ではないことを示すために、arg[0-5].V=2に変更する。さらに、%o0-1/%i0-1について、rti[0-1].V=1に変更し、rti[0-1].Valに書き込みデータを記録する。その後、さらにRWをたどり、該RBが関数の場合、処理を終了する。一方、該RBが関数ではない(ループである)場合、arg[0-1].V=0であれば、以降の読み出しは入力ではないことを示すために、arg[0-1].V=2に変更し、rti[0-1].V=1に変更し、rti[0-1].Valに書き込みデータを記録し、処理を終了する。

[0405] 一方、(2-2)該RBがループの場合、(a)%g0-7でgrr[0-7].V=0であれば、grr[0-7].V=2に変更し、grr[0-7].Valに書き込みデータを記録し、処理を終了する。(b)%o0-7でarg[0-7].V=0であれば、arg[0-7].V=2に変更し、arg[0-7].Valに書き込みデータを記録し、処理を終了する。(c)%l0-7でlrr[0-7].V=0であれば、lrr[0-7].V=2に変更し、lrr[0-7].Valに書き込みデータを記録し、処理を終了する。(d)%i0-7でirr[0-7].V=0であれば、irr[0-7].V=2に変更し、irr[0-7].Valに書き込みデータを記録し、次のRWエントリに進む。

[0406] (3)浮動小数点レジスタREADの場合

まず、RWのTOPからBOTTOMまで順にたどる。そして(3-1)該RBが関数の場

合、何もせずに処理を終了する。一方、(3−2)該RBがループの場合、frr[0−31].V=0であれば、frr[0−31].V=1に変更し、frr[0−31].Valに読み出しデータを記録し、処理を終了する。

[0407] (4)浮動小数点レジスタWRITEの場合

まず、RWのTOPからBOTTOMまで順にたどる。そして(4−1)該RBが関数かつ%f0−1の場合、rtf[0−1].V=1に変更し、rtf[0−1].Valに書き込みデータを記録する。さらにRWをたどり、frr[0−1].V=0であれば、以降の読み出しは入力ではないことを示すために、frr[0−1].V=2に変更し、rtf[0−1].V=1に変更し、rtf[0−1].Valに書き込みデータを記録し、処理を終了する。

[0408] 一方、(4−2)該RBがループの場合、frr[0−31].V=0であれば、frr[0−31].V=2に変更し、frw[0−31].V=1に変更し、frw[0−7].Valに書き込みデータを記録し、処理を終了する。

[0409] (5)条件コードレジスタICC−READの場合

まず、RWのTOPからBOTTOMまで順にたどる。そして(5−1)該RBが関数の場合、何もせずに処理を終了する。一方、(5−2)該RBがループの場合、icr.V=0であれば、icr.V=1に変更し、icr.Valに読み出しデータを記録し、処理を終了する。

[0410] (6)条件コードレジスタICC−WRITEの場合

まず、RWのTOPからBOTTOMまで順にたどる。そして(6−1)該RBが関数の場合、何もせずに処理を終了する。一方、(6−2)該RBがループの場合、icr.V=0であれば、icr.V=2、icw.V=1に変更し、icw.Valに書き込みデータを記録し、処理を終了する。

[0411] (7)浮動小数点条件コードレジスタFCC−READの場合

まず、RWのTOPからBOTTOMまで順にたどる。そして(7−1)該RBが関数の場合、何もせずに処理を終了する。一方、(7−2)該RBがループの場合、fcr.V=0であれば、fcr.V=1に変更し、fcr.Valに読み出しデータを記録し、処理を終了する。

[0412] (8)条件コードレジスタICC−WRITEの場合

まず、RWのTOPからBOTTOMまで順にたどる。そして(8−1)該RBが関数の場合、何もせずに処理を終了する。一方、(8−2)該RBがループの場合、fcr.V=0であ

れば、fcr.V=2、fcw.V=1に変更し、fcw.Valに書き込みデータを記録し、処理を終了する。

[0413] (9)主記憶READの場合

まず、RWのTOPからBOTTOMまで順にたどる。そして、RBにWRITEデータとして登録済である場合は、その値を使用する。一方、上記の場合ではなく、RBにREADデータとして登録済である場合には、その値を使用する。さらに、いずれにも登録済でない場合は、キャッシュを経由して主記憶3から読み込む。

[0414] その後、再度RWのTOPからBOTTOMまで順にたどる。そして、(a)アドレスが、RBに登録されているsp+64の場合、構造体ポインタの読み出しであるので、arg0.V=0であれば、arg0.V=1に変更し、arg0.Valに読み出しデータを記録する。(b)上記の(a)の場合でなく、アドレスが、LIMIT以上sp+92未満であれば、登録不要領域であるので、何もしない。(c)上記の(b)の場合でない場合、WRITEデータとして登録済であるかどうかを検査し、そうであれば、すでに上書きされたあとのREADであるので登録不要であり、何もしない。(d)上記の(c)でない場合、READデータとして登録済であるかどうかを検査し、そうであれば、すでに登録済であるので登録不要であり、何もしない。(e)上記の(d)でない場合、READデータとしての登録が必要であるので、RFに主記憶READアドレスを確保し、READデータとして登録する。RFに主記憶アドレスを確保できなかった場合には、登録不能であるため、そのRWエントリからBOTTOMまでに対応するRBエントリを全て無効化する。

[0415] (10)主記憶WRITEの場合

まず、キャッシュを経由して、主記憶3に書き込む。そして、ベースレジスタが14(%sp)かつオフセットが92以上である場合、引数の第7ワードを検出したことを記憶する。

[0416] その後、RWのTOPからBOTTOMまで順にたどる。そして、(a)アドレスが、RBに登録されているsp+64の場合、構造体ポインタの読み出しであるので、arg0.V=0であれば、arg0.V=2に変更する。(b)上記の(a)の場合ではなく、アドレスがLIMIT以上sp+92未満であれば、登録不要領域であるので、何もしない。(c)上記の(b)の場合でない場合、WRITEデータとして登録済であるかどうかを検査し、そうであれば、す

でにアドレスは登録済であるので、内容を新しいWRITEデータに更新する。(d)上記の(c)でない場合、WRITEデータとしての登録が必要であるので、RFに主記憶WRITEアドレスを確保し、WRITEデータとして登録する。RFに主記憶アドレスを確保できなかった場合には、登録不能であるため、そのRWエントリからBOTTOMまでに対応するRBエントリを全て無効化する。

[0417] (ループを含む多重再利用)

1レベルで上記のような再利用機構を用いた場合、図46(a)に示した例で言えば、リーフ関数としての関数Bや、関数Bの内部にあるループCなどをそれぞれ再利用することが可能となる。これに対して、ある関数を一度実行しただけで、その関数の内部に含まれる関数やループを含む全ての命令区間が再利用可能となるように登録を行う仕組みが多重再利用である。例えば上記の例で言えば、多重再利用によれば、関数Aを一度実行しただけで、入れ子関係にあるA, B, Cの全ての命令区間が再利用可能となる。以下に、多重再利用を実現する上で必要とされる機能拡張について説明する。

[0418] 関数Aおよび関数Dの概念的な構造については、前記した図6に示したとおりである。また、図6に示す関数A, DおよびループB, C, E, Fの入れ子構造において、内側の構造のレジスタ入出力(太枠セル領域)が、外側の構造のレジスタ入出力となる影響範囲(矢印)についても、前記した図7に示したとおりである。

[0419] 以上のことから、多重再利用を実現するには、前述したRFおよびRBを関数やループの入れ子構造と関連づける機構が必要である。図30に示すように、再利用ウィンドウ(RW)を装備することによって、現在実行中かつ登録中であるRFおよびRBの各エントリ(図中ではA、B、Cと示す)をスタック構造として保持する。関数やループの実行中は、RWに登録されている全てのエントリについて、これまでに述べた方法に基づいて、レジスタおよび主記憶参照を登録していく。

[0420] この際に、あるエントリに関して、(1)登録可能項目数の超過、(2)引数の第7ワードの検出、(3)システムコールの検出、によって再利用不可能であると判断した場合には、RWを用いて、そのエントリに対応するRBおよび上位のRBを特定し、登録を中止することができる。

[0421] なお、RWの深さは有限であるものの、一度に登録可能な多重度を超えて関数やループを検出した場合には、外側の命令区間から順次登録を中止し、より内側の命令区間を登録対象に加えることによって、入れ子関係の動的変化に追随することができる。また、実行および登録中(例えばA)に、再利用可能な命令区間(例えばD)に遭遇した場合には、登録済の入出力をそのまま登録中エントリに追加することによって、RWの深さを超えるAの多重再利用も可能となる。

[0422] (並列事前実行)

実施の形態1において示したように、多重再利用を行うプロセッサとしてのMSP1Aとは別に、命令区間の事前実行によってRBエントリを有効にするプロセッサとしてのSSP1Bを複数個設けることによって、さらなる高速化を図ることができる。

[0423] 並列事前実行機構を行うためのハードウェア構成は、前記した図25に示すような構成となる。同図に示すように、RW4A・4B、演算器5A・5B、レジスタ6A・6B、キャッシュ7A・7Bは、各プロセッサごとに独立して設けられている一方、RF／RB2'、および主記憶3は全てのプロセッサが共有するようになっている。同図において、破線は、MSP1AおよびSSP1BがRF／RB2'に対して入出力を登録するパスを示している。

[0424] ここで、並列事前実行を実現する上での課題は、(1)どのように主記憶一貫性を保つか、(2)どのように入力を予測するかが挙げられる。以下に、これらの課題に対する解決手法について説明する。

[0425] (主記憶一貫性に関する課題の解決方法)

まず、上記の課題(1)どのように主記憶一貫性を保つかについて説明する。特に予測した入力パラメータに基づいて命令区間を実行する場合、主記憶3に書き込む値がMSP1AとSSP1Bとで異なることになる。これを解決するために、図25に示すように、SSP1Bは、RBへの登録対象となる主記憶参照にはRF／RB2'、また、その他の局所的な参照にはSSP1Bごとに設けた局所メモリとしてのLocal7Bを使用することとし、Cache7Bおよび主記憶3への書き込みを不要としている。なお、MSP1Aが主記憶3に対して書き込みを行った場合には、対応するSSP1Bのキャッシュラインが無効化される。

[0426] 具体的には、RBへの登録対象のうち、読み出しが先行するアドレスについては主記憶3を参照し、MSP1Aと同様にアドレスおよび値をRBへ登録する。以後、主記憶3ではなくRBを参照することによって、他のプロセッサからの上書きによる矛盾の発生を避けることができる。局所的な参照については、読み出しが先行するということは、変数を初期化せずに使うことに相当し、値は不定でよいことになるので、主記憶3を参照する必要はない。

[0427] なお、局所メモリとしてのLocal7Bの容量は有限であり、関数フレームの大きさがLocal7Bの容量を超えた場合など、実行を継続できない場合は、事前実行を打ち切るようにする。また、事前実行の結果は主記憶3に書き込まれないので、事前実行結果を使って、さらに次の事前実行を行うことはできない。

[0428] (予測機構)

次に、上記の課題(2)どのように入力を予測するかについて説明する。事前実行に際しては、RBの使用履歴に基づいて将来の入力を予測し、SSP1Bへ渡す必要がある。このために、RF／RB2'には、予測処理部2Bが設けられている。この予測処理部2Bは、RFの各エントリごとに設けた小さなプロセッサによって構成され、MSP1AやSSP1Bとは独立して入力予測値を求めるものである。

[0429] 前記したように、従来の入力予測では、RBにおける入力側に登録された全てのアドレスが一律に扱われたことによって、予測の的中率を下げる結果となっている。この問題を解決するためには、予測が的中する可能性が高いアドレスと、予想が外れる可能性が高いアドレスを区別するとともに、値の変化にも着目して必要最小限のアドレスのみを予測対象とすることが必要である。

[0430] 予測が的中することが期待できるアドレスとは、アドレスが固定しており、かつ、値が単調変化するアドレスである。このようなアドレスには、ラベルによって参照される帯域変数、および、スタックポインタやフレームポインタをベースレジスタとして参照される局所変数(フレーム内変数)などがある。

[0431] これらのアドレスを識別するために、ロード命令実行時のアドレス計算が参照するレジスタに定数フラグ(Const-FLAG)が設けられる。スタックポインタやフレームポインタとして用いるレジスタについては無条件に定数フラグがセットされるものとする。その

他のレジスタについては、定数をセットする命令が実行された時に定数フラグ(Const-FLAG)がセットされるものとする。

[0432] 次に、過去に参照したアドレスのうち、一度も書き込みが行われないアドレスについては、内容が変化していないことが保証されることになり、このようなアドレスについては予測する必要がないことになる。よって、このようなアドレスを区別するために、書き込みが行われたことを示す変更フラグ(C-FLAG)が設けられる。入力要素としてのアドレスをRF／RBに新規に記録する時には、該アドレスに対応する変更フラグ(C-FLAG)がリセットされ、登録後に該アドレスに対してストア命令が実行された時に、変更フラグ(C-FLAG)がセットされる。

[0433] また、入力要素としてのアドレスを履歴保存対象とするか否かを示す履歴マスク(P-Mask)が設けられる。入力要素としてのアドレスをRF／RBに新規に記録する時には、該アドレスに対応する履歴マスク(P-Mask)(履歴フラグ)がリセットされる。そして、ロード命令実行時に、該アドレスを生成したレジスタに対応する定数フラグ(Const-FLAG)がセットされている場合には、履歴マスク(P-Mask)のうちロード対象となったバイト位置がセットされる。

[0434] 以上の定数フラグ(Const-FLAG)、変更フラグ(C-FLAG)、および履歴マスク(P-Mask)の設定の制御は、RF／RB2'に設けられているRB登録処理部2Aによって行われる。このRB登録処理部2Aは、小さなプロセッサによって構成され、上記のような判断を行うことによって定数フラグ(Const-FLAG)、変更フラグ(C-FLAG)、および履歴マスク(P-Mask)の設定を行う。

[0435] (命令区間の実行例)

ここで、命令区間の一例として、図49に示す命令区間が、図24に示したRFおよびRBの構成によって実行された場合の例について説明する。同図において、PCは、該命令区間が開始された際のPC値を示している。すなわち、命令区間の先頭が1000番地となっている。また、図31は、図49に示す命令区間が実行された場合のRBにおける実際の登録状況を示している。

[0436] 第1の命令において、アドレス定数A1がレジスタR0にセットされる。この命令は、定数をセットする命令であるので、レジスタR0に対応する定数フラグ(Const-FLAG)が

セットされる。

[0437] 第2の命令において、レジスタR0の内容をアドレスとする主記憶3からロードされた4バイトデータ(00110000)がレジスタR1に格納される。この場合、アドレスA1、マスク(FFFFFFFF)、データ(00110000)は、入力としてRBにおけるInput側の第1列に登録され、レジスタ番号R1、マスク(FFFFFFFF)、およびデータ(00110000)は出力としてRBにおけるOutput側の第1列に登録される。

[0438] また、アドレスとして用いたレジスタR0に対応する定数フラグ(Const-FLAG)がセットされているので、アドレスA1に対応する履歴マスク(P-Mask)がセットされる。ここで、対象となるデータは(00110000)の4バイトデータであるので、これに対応して、アドレスA1に対応する履歴マスク(P-Mask)には(FFFFFFFF)がセットされる。そして、レジスタR1は、定数がセットされるものではないことになるので、レジスタR1に対応する定数フラグ(Const-FLAG)はリセットされる。

[0439] 第3の命令において、アドレス定数A2がレジスタR0にセットされる。この命令は、定数をセットする命令であるので、レジスタR0に対応する定数フラグ(Const-FLAG)がセットされる。

[0440] 第4の命令において、レジスタR0の内容をアドレスとする主記憶3からロードされた1バイトデータ(02)がレジスタR2に格納される。この場合、アドレスA2、マスク(FF000000)、およびデータ(02)は入力としてRBにおけるInput側の第2列に登録される。この際、アドレスA2の残り3バイトについては、Don't Careを意味する「-」が格納される。レジスタ番号R2、マスク(FFFFFFFF)およびデータ(00000002)は出力としてRBにおけるOutput側の第2列に登録される。

[0441] また、アドレスとして用いたレジスタR0に対応する定数フラグ(Const-FLAG)がセットされているので、アドレスA2に対応する履歴マスク(P-Mask)がセットされる。ここで、対象となるデータは(02)の1バイトデータであるので、これに対応して、アドレスA2に対応する履歴マスク(P-Mask)には(FF000000)がセットされる。そして、レジスタR2は、定数がセットされるものではないことになるので、レジスタR2に対応する定数フラグ(Const-FLAG)はリセットされる。

[0442] 第5の命令において、アドレス(A2+R2)からロードされた1バイトデータ(22)がレ

ジスタR2に格納されている。アドレスR2の値は(02)であったので、アドレス(A2+02)、およびデータ(22)が、入力としてRBにおけるInput側の第2列に追加登録される。この際、アドレス(A2+02)の部分に登録が行われ、アドレス(A2+01)および(A2+03)に対応する部分は、Don't Careを意味する「−」のままとなる。すなわち、アドレスA2に対応するマスクは(FF00FF00)となる。レジスタ番号R2、マスク(FFFFFFFF)、およびデータ(00000022)は、出力としてRBにおけるOutput側の第2列に上書きされる。

[0443] また、アドレスとして用いたレジスタR2に対応する定数フラグ(Const-FLAG)がリセットされているので、アドレス(A2+02)に対応する履歴マスク(P-Mask)はセットされない。すなわち、アドレスA2に対応する履歴マスク(P-Mask)は(FF000000)のままとなる。そして、レジスタR2は、定数がセットされるものではないことになるので、レジスタR2に対応する定数フラグ(Const-FLAG)はリセットされる。

[0444] 第6の命令において、アドレス定数A3がレジスタR0にセットされる。この命令は、定数をセットする命令であるので、レジスタR0に対応する定数フラグ(Const-FLAG)がセットされる。

[0445] 第7の命令において、レジスタR0の内容をアドレスとする主記憶3からロードされた1バイトデータ(33)がレジスタR3に格納される。この場合、アドレスA3、マスク(00FF0000)、およびデータ(33)は入力としてRBにおけるInput側の第3列に登録される。レジスタ番号R3、マスク(FFFFFFFF)、およびデータ(00000033)は出力としてRBにおけるOutput側の第3列に登録される。

[0446] また、アドレスとして用いたレジスタR0に対応する定数フラグ(Const-FLAG)がセットされているので、アドレスA3に対応する履歴マスク(P-Mask)がセットされる。ここで、対象となるデータは(33)の1バイトデータであるので、これに対応して、アドレスA3に対応する履歴マスク(P-Mask)には(00FF0000)がセットされる。そして、レジスタR3は、定数がセットされるものではないことになるので、レジスタR3に対応する定数フラグ(Const-FLAG)はリセットされる。

[0447] 第8の命令において、アドレス(R1+R2)からロードされた1バイトデータ(44)がレジスタR4に格納される。この場合、アドレスR1とアドレスR2は命令区間の内部にて上

書きされたレジスタのアドレスとなるので、アドレスR1およびアドレスR2は命令区間の入力とはならない。一方、アドレス(R1+R2)によって生成されたアドレスA4は命令区間の入力であるので、アドレスA4、マスク(00FF0000)、およびデータ(44)は入力としてRBにおけるInput側の第4列に登録される。レジスタ番号R4、マスク(FFFFFFFF)、およびデータ(00000044)は出力としてRBにおけるOutput側の第4列に登録される。

[0448] また、アドレスとして用いたレジスタR1およびレジスタR2に対応する定数フラグ(Const-FLAG)がリセットされているので、アドレスA4に対応する履歴マスク(P-Mask)はセットされない。すなわち、アドレスA4に対応する履歴マスク(P-Mask)は(00000000)となる。そして、レジスタR4は、定数がセットされるものではないことになるので、レジスタR4に対応する定数フラグ(Const-FLAG)はリセットされる。

[0449] 第9の命令において、レジスタR5から値が読み出され、読み出された値に1が加えられた結果が再びレジスタR5に格納される。この場合、レジスタR5、マスク(FFFFFFFF)、およびデータ(00000100)は入力としてRBにおけるInput側の第5列に登録される。また、レジスタ番号R5、マスク(FFFFFFFF)、およびデータ(00000101)は出力としてRBにおけるOutput側の第5列に登録される。この時、レジスタR5は、定数がセットされるものではないことになるので、レジスタR5に対応する定数フラグ(Const-FLAG)はリセットされる。

[0450] その後、アドレスA2、およびアドレスA3に対してストア命令が実行され、アドレスA2、およびアドレスA3に対して変更フラグ(C-FLAG)がセットされたとする。

[0451] 以上の結果、変更フラグ(C-FLAG)がセットされ、かつ、履歴マスク(P-Mask)がセットされたマスク位置は、アドレスA2の第1バイト、アドレスA3の第2バイトのみとなる。このマスク位置のみに対応するアドレス、マスク、および値が、予測対象として、命令区間ごとに過去の入力履歴を保持する履歴情報として、RBのエントリに記録される。また、RBの入力パターンに登録されたレジスタについては無条件に予測対象として履歴として記録される。

[0452] 図32は、図49に示す命令区間が繰り返し実行された場合における、履歴としてRBに登録された例を示している。同図に示すように、RBには、アドレスA2の列に履歴

マスク(P-Mask)として(FF000000)、アドレスA3の列に履歴マスク(P-Mask)として(00FF0000)、およびアドレスR5の列に履歴マスク(P-Mask)として(FFFFFFFF)が記憶される。そして、Timeが1ー4に変化する間に、各アドレスにおける履歴マスク(P-Mask)に対応する値が変化することになる。各履歴の間に示されるdiffは、対応する入力要素の値の変化量(差分)を示している。このdiffは、予測処理部2Bによって算出される。

[0453] 同図に示す例では、アドレスA2およびアドレスR5の列に関しては、Timeが1ー4に変化する間におけるdiffが全て01となっている。よって、これらのアドレスに対応する値は、単位時間あたりに01ずつ増加していくことが予想される。一方、アドレスA3の列に関しては、Timeが1ー4に変化する間に、diffは00であったり02であったりしている。したがって、アドレスA3に関しては、予測することが困難であることがわかる。

[0454] 以上より、予測処理部2Bは、履歴において、差分が一定となっているアドレスに関して、該差分がその後も継続するものと仮定して予測を行うとともに、差分が一定でない、または差分が0となっているアドレスに関しては予測を行わないようにする。

[0455] 図33は、上記の予測に基づいて、予測処理部2BがアドレスA2およびアドレスR5の値に関して予測を行った場合の、予測エントリとしてRBに記録される入力要素の状態を示している。同図において、アドレス(A2+4)およびアドレスA3に関しては、予測値を求めずに直接主記憶3を参照することによって得られたものとなっている。

[0456] このように入力要素の予測値が算出されると、SSP1Bが、この予測入力要素に基づいて命令区間を実行することによって出力要素が算出され、この予測出力要素が予測エントリとしてRBに記憶される。その後、MSP1Aによって命令区間が実行され、予測エントリとしてRBに記憶されている予測入力要素と同じ入力値が入力された場合に、それに対応する予測出力要素を出力することによって再利用が実現されることになる。

[0457] (RF／RBの第2の構成例)

次に、RF／RB2'の第2の構成例について説明する。第2の構成例としてのRF／RB2'は、実施の形態1において図1として示した命令区間記憶部2と同様の構成となっている。すなわち、第2の構成例としてのRF／RB2'は、RB、RF、RO1(第2出

力パターン記憶手段)、およびRO2(第1出力パターン記憶手段)を備えた構成となっている。各構成および動作については前記した内容と同様であるので、ここではその説明を省略する。

[0458] (第2の構成例における予測機構)

第2の構成例では、命令区間の実行時における入出力パターンを一時的に格納する場所は、RW4A・4Bとなる。ここで、前記した第1の構成例では、命令区間の実行時における入出力パターンはRBに直接登録されていたので、RW4A・4BはRBの各行に対するポインタによって実現されていた。これに対して、第2の構成例では、RFおよびRBが木構造によって構成されているので、RW4A・4Bが直接RBの行をポイントすることができない。すなわち、第2の構成例では、RW4A・4Bは、RBの各行に対するポインタとして機能するものではなく、命令区間の実行時における入出力パターンを一時的に格納する実質的なメモリとして機能することになる。

[0459] また、第2の構成例においても、所定の命令区間が繰り返し実行された場合における入力パターンの履歴エントリを格納する一時格納メモリ領域として、図24に示すようなRFおよびRBが設けられている。ただし、この場合には、RBにおけるエントリの行は、履歴エントリを格納する履歴格納行としての数行によって構成されることになる。

[0460] 命令区間が実行されると、その入力要素がRW4A・4Bに順次格納され、全ての入力要素が揃い、演算が行われることによって出力要素が確定すると、この入出力パターンが、上記履歴格納行に格納されるとともに、上記のような木構造の入出力パターン格納機構に格納されることになる。

[0461] また、所定の命令区間が繰り返し実行された場合には、履歴格納行に順次格納され、所定の数の履歴が格納された時点で、上記のように予測処理部2Bによって予測が行われ、予測に基づいてSSP1Bによって実行された結果は、上記のような木構造の入出力パターン格納機構に格納されることになる。

[0462] <実施の形態3>

本発明のさらに他の実施形態について図面に基づいて説明すると以下の通りである。

[0463] (データ処理装置の構成)

本実施形態に係るデータ処理装置の概略構成を図35に示す。同図に示すように、該データ処理装置は、MSP1A、SSP1B、再利用表としての命令区間記憶部(入出力記憶手段)2、および主記憶(主記憶手段)3を備えた構成となっており、主記憶3に記憶されているプログラムデータなどを読み出して各種演算処理を行い、演算結果を主記憶3に書き込む処理を行うものである。なお、同図に示す構成では、SSP1Bを1つ備えた構成となっているが、2つ以上備えた構成となっていてもよい。

[0464] 命令区間記憶部2は、プログラムにおける関数およびループを再利用するためのデータを格納するメモリ手段であり、RF、RB、RB登録処理部(登録処理手段)2A、および予測処理部(予測処理手段)2Bを備えた構成となっている。この命令区間記憶部2におけるRFおよびRBの詳細、ならびにRB登録処理部2Aおよび予測処理部2Bの詳細については後述する。

[0465] 主記憶3は、MSP1AおよびSSP1Bの作業領域としてのメモリであり、例えばRAMなどによって構成されるものである。例えばハードディスクなどの外部記憶手段からプログラムやデータなどが主記憶3に読み出され、MSP1AおよびSSP1Bは、主記憶3に読み出されたデータに基づいて演算を行うことになる。

[0466] MSP1Aは、RW(再利用記憶手段)4A、演算器(第1の演算手段)5A、レジスタ6A、Cache7A、および通信部9Aを備えた構成となっている。また、SSP1Bは、RW(再利用記憶手段)4B、演算器(第2の演算手段)5B、レジスタ6B、Cache／Local7B、判定部8B、および通信部9Bを備えた構成となっている。

[0467] RW4A・4Bは、再利用ウィンドウであり、現在実行中かつ登録中であるRFおよびRBの各エントリをリング構造のスタックとして保持するものである。このRW4A・4Bは、実際のハードウェア構造としては、命令区間記憶部2における特定のエントリをアクティブにする制御線の集合によって構成される。

[0468] 演算器5A・5Bは、レジスタ6A・6Bに保持されているデータに基づいて演算処理を行うものであり、ALUと呼ばれるものである。レジスタ6A・6Bは、演算器5A・5Bによって演算を行うためのデータを保持する記憶手段である。なお、本実施形態では、演算器5A・5B、およびレジスタ6A・6Bは、SPARCアーキテクチャに準じたものとする。Cache7A・7Bは、主記憶3と、MSP1AおよびSSP1Bとの間でのキャッシュメモリ

として機能するものである。なお、SSP1Bでは、Cache7Bには、局所メモリとしてのLocal7Bが含まれているものとする。

[0469] 判定部8Bは、後述する事前実行の起動開始後の主記憶読み出しが行われる際に、RBにおける入出力記録行(後述)、予測値格納領域(後述)、待機用アドレス格納領域(後述)、およびCache／Local7Bのうち、どこから値を読み出すかを判定するブロックである。この判定処理の詳細については後述する。この判定部8Bは、SSP1B内に設けられた小さなプロセッサによって実現される。

[0470] 通信部9A・9Bは、MSP1AまたはSSP1Bによって主記憶書き込みが行われる場合に、その旨をその他の全てのSSP1B…またはMSP1Aに対して通知するブロックである。この通信部9A・9Bは、MSP1AまたはSSP1B内に設けられた小さなプロセッサによって実現される。

[0471] (RF／RBの構成)

図34は、本実施形態における命令区間記憶部2におけるRFおよびRBの構成の概要を示している。同図に示すように、RFは、複数のエントリを格納しており、各エントリに対して、該エントリが有効であるか否かを示すV、エントリ入れ替えのヒントを示すLRU、関数の先頭アドレスを示すStart、参照すべき主記憶アドレスを示すRead/Write、および、関数とループとを区別するF/Lを保持している。

[0472] また、RBは、RFに格納されているエントリに対応して複数のエントリを格納しており、各エントリに対して、該エントリが有効であるか否かを示すV、エントリ入れ替えのヒントを示すLRU、関数またはループを呼び出す際の直前のスタックポイント%spを示すSP、引数(Args.)(V:有効エントリ、Val.:値)、主記憶値(C-FLAG:Readアドレスの変更フラグ、P-Mask:Readアドレスの履歴マスク、Mask:Read/Writeアドレスの有効バイト、Value:値、S-Count:Read/Writeアドレスのストアカウンタ)、返り値(Return Values)(V:有効エントリ、Val.:値)、ループの終了アドレス(End)、ループ終了時の分岐方向を示すtaken/not、および、引数や返り値以外のレジスタおよび条件コード(Regs.,CC)を保持している。また、RBは、1つ以上のレジスタアドレスに対応して定数フラグ(Const-FLAG)を格納するメモリ領域を保持している。なお、定数フラグ(Const-FLAG)の詳細については後述する。

[0473] 上記のRFおよびRBにおける各項目についてより詳細に説明する。上記Vは、上記のようにエントリが有効であるか否かを示すものであるが、具体的には、未登録時には「0」、登録中である場合には「2」、登録済である場合には「1」の値が格納されるようになっている。例えば、RFまたはRBを確保する際に、未登録エントリ(V=0)があれば、これを使用し、未登録エントリがなければ、登録済エントリ(V=1)の中からLRUが最小のものを選択して上書きすることになる。登録中エントリ(V=2)は使用中であるので上書きすることはできない。

[0474] 上記LRUは、一定時間間隔で右へシフトされていくシフトレジスタの中の「1」の個数を示したものである。RFの場合、このシフトレジスタは、該当エントリに関して、再利用のための登録を行ったか、もしくは再利用を試みた場合に、左端に「1」が書き込まれるようになっている。したがって、該当エントリが頻繁に使用されれば、LRUは大きな値となり、一定期間使用されなければ、LRUの値は0となる。一方、RBの場合、シフトレジスタには、該当エントリが再利用された場合に「1」が書き込まれるようになっている。したがって、該当エントリが頻繁に使用されれば、LRUは大きな値となり、一定期間使用されなければ、LRUの値は0となる。

[0475] 上記RBにおける主記憶値のMaskについて説明する。一般に、アドレスとデータとを1バイトずつ管理することにすれば管理が可能であるが、実際には、4バイト単位でデータを管理する方がキャッシュ参照を高速に行うことができる。そこで、RFでは、主記憶アドレスを4の倍数で記憶するようになっている。一方、管理単位を4バイトとする場合、1バイト分だけをロードすることに対応できるようにするために、4バイトのうちでどのバイトが有効であるかを示す必要がある。すなわち、Maskは、4バイトのうちでどのバイトが有効であるかを示す4ビットのデータとなっている。例えば、C001番地から1バイト分をロードした結果、値がE8であった場合、RFには、アドレスC000が登録され、RBのMaskに「0100」、Valueに「00E80000」が登録されることになる。なお、Readアドレスにおける変更フラグ(C-FLAG)および履歴マスク(P-Mask)、ならびにRead/Writeアドレスにおけるストアカウンタ(S-Count)の詳細については後述する。

[0476] 上記の引数や返り値以外のレジスタおよび条件コード(Regs.,CC)について説明する。本実施形態では、SPARCアーキテクチャレジスタのうち、汎用レジスタ%g0-7、

%o0-7、%l0-7、%i0-7、浮動小数点レジスタ%f0-31、条件コードレジスタICC、浮動小数点条件コードレジスタFCCを用いるようになっている(詳細は後述する)。このうち、リーフ関数の入力は汎用レジスタ%o0-5、出力は汎用レジスタ%o0-1、また、非リーフ関数の入力は汎用レジスタ%i0-5、出力は汎用レジスタ%i0-1、になり、入力は、arg[0-5]、出力は、rti[0-1]に登録される。SPARC—ABIの規定では、これら以外のレジスタは関数の入出力にはならないので、関数に関してはRBにおける引数(Args.)の項で十分である。

[0477] 一方、SPARC—ABIの規定では、ループの入出力に関しては、用いられるレジスタの種類を特定することはできないので、ループの入出力を特定するには、全ての種類のレジスタに関してRBに登録する必要がある。よって、RBにおけるRegs.,CCには、%g0-7、%o0-7、%l0-7、%i0-7、%f0-31、ICC、FCCが登録されるようになっている。

[0478] 以上のように、命令区間記憶部2において、Readアドレスは RFが一括管理し、MaskおよびValueはRBが管理している。これにより、Readアドレスの内容とRBの複数エントリをCAMによって一度に比較する構成を可能としている。

[0479] なお、図35に示すように、本実施形態におけるRBには、入出力記録行(入出力記録領域)、区間毎情報として履歴格納行(履歴格納領域)、予測値格納領域、および待機要アドレス格納領域、ならびに予測実行結果記録行が設けられている。これらの入出力記録行、履歴格納行、予測値格納領域、および待機要アドレス格納領域、ならびに予測実行結果記録行は、図34に示すRBにおけるエントリにほぼ準じた形式で実現されるが、それぞれ格納形式が若干異なっている。これらの格納形式の詳細については後述する。

[0480] (再利用処理の概略)

関数およびループのそれぞれの場合についての再利用処理の概略については、実施の形態2における(再利用処理の概略)において説明した内容と同様であるので、ここではその説明を省略する。

[0481] (命令区間の実行時における処理の流れ)

命令がデコードされた場合の具体的な処理の流れについても、実施の形態2における(命令区間の実行時における処理の流れ)において説明した内容と同様である

ので、ここではその説明を省略する。また、命令がデコードされた結果、関数呼び出し命令である場合、関数復帰命令である場合、後方分岐成立の場合、後方分岐不成立の場合、およびその他の命令の場合について、それぞれ処理の流れについても、前記した実施の形態2において説明した内容と同様である。

[0482] (ループを含む多重再利用)

ループを含む多重再利用についても、実施の形態2における(ループを含む多重再利用)において説明した内容と同様であるので、ここではその説明を省略する。

[0483] (並列事前実行)

前記したように、多重再利用を行うプロセッサとしてのMSP1Aとは別に、命令区間の事前実行によってRBエントリを有効にするプロセッサとしてのSSP1Bを複数個設けることによって、さらなる高速化を図ることができる。

[0484] 並列事前実行機構を行うためのハードウェア構成は、前記した図35に示すような構成となる。同図に示すように、RW4A・4B、演算器5A・5B、レジスタ6A・6B、キャッシュ7A・7Bは、各プロセッサごとに独立して設けられている一方、命令区間記憶部2、および主記憶3は全てのプロセッサが共有するようになっている。

[0485] ここで、並列事前実行を実現する上での課題は、(1)どのように主記憶一貫性を保つか、(2)どのように入力を予測するかが挙げられる。以下に、これらの課題に対する解決手法について説明する。

[0486] (主記憶一貫性に関する課題の解決方法)

まず、上記の課題(1)どのように主記憶一貫性を保つかについて説明する。特に予測した入力パラメータに基づいて命令区間を実行する場合、主記憶3に書き込む値がMSP1AとSSP1Bとで異なることになる。これを解決するために、図35に示すように、SSP1Bは、RBへの登録対象となる主記憶参照には命令区間記憶部2、また、その他の局所的な参照にはSSP1Bごとに設けた局所メモリとしてのLocal7Bを使用することとし、Cache7Bおよび主記憶3への書き込みを不要としている。なお、MSP1Aが主記憶3に対して書き込みを行った場合には、対応するSSP1Bのキャッシュラインが無効化される。

[0487] 具体的には、RBへの登録対象のうち、読み出しが先行するアドレスについては主

記憶3を参照し、MSP1Aと同様にアドレスおよび値をRBへ登録する。以後、主記憶3ではなくRBを参照することによって、他のプロセッサからの上書きによる矛盾の発生を避けることができる。局所的な参照については、読み出しが先行するということは、変数を初期化せずに使うことに相当し、値は不定でよいことになるので、主記憶3を参照する必要はない。

[0488] なお、局所メモリとしてのLocal7Bの容量は有限であり、関数フレームの大きさがLocal7Bの容量を超えた場合など、実行を継続できない場合は、事前実行を打ち切るようにする。また、事前実行の結果は主記憶3に書き込まれないので、事前実行結果を使って、さらに次の事前実行を行うことはできない。

[0489] (予測機構の参考例)

次に、上記の課題(2)どのように入力を予測するかについて説明する。事前実行に際しては、RBの使用履歴に基づいて将来の入力を予測し、SSP1Bへ渡す必要がある。このために、命令区間記憶部2には、予測処理部2Bが設けられている。この予測処理部2Bは、RFの各エントリごとに設けた小さなプロセッサによって構成され、MSP1AやSSP1Bとは独立して入力予測値を求めるものである。

[0490] 前記したように、従来の入力予測では、RBにおける入力側に登録された全てのアドレスが一律に扱われたことによって、予測の的中率を下げる結果となっている。この問題を解決するためには、予測が的中する可能性が高いアドレスと、予想が外れる可能性が高いアドレスを区別するとともに、値の変化にも着目して必要最小限のアドレスのみを予測対象とすることが必要である。

[0491] 予測が的中することが期待できるアドレスとは、アドレスが固定しており、かつ、値が単調変化するアドレスである。このようなアドレスには、ラベルによって参照される帯域変数、および、スタックポインタやフレームポインタをベースレジスタとして参照される局所変数(フレーム内変数)などがある。

[0492] これらのアドレスを識別するために、ロード命令実行時のアドレス計算が参照するレジスタに定数フラグ(Const-FLAG)が設けられる。スタックポインタやフレームポインタとして用いるレジスタについては無条件に定数フラグがセットされるものとする。その他のレジスタについては、定数をセットする命令が実行された時に定数フラグ(

Const-FLAG)がセットされるものとする。

[0493] 次に、過去に参照したアドレスのうち、一度も書き込みが行われないアドレスについては、内容が変化していないことが保証されることになり、このようなアドレスについては予測する必要がないことになる。よって、このようなアドレスを区別するために、書き込みが行われたことを示す変更フラグ(C-FLAG)が設けられる。入力要素としてのアドレスをRF／RBに新規に記録する時には、該アドレスに対応する変更フラグ(C-FLAG)がリセットされ、登録後に該アドレスに対してストア命令が実行された時に、変更フラグ(C-FLAG)がセットされる。

[0494] また、入力要素としてのアドレスを履歴保存対象とするか否かを示す履歴マスク(P-Mask)が設けられる。入力要素としてのアドレスをRF／RBに新規に記録する時には、該アドレスに対応する履歴マスク(P-Mask)(履歴フラグ)がリセットされる。そして、ロード命令実行時に、該アドレスを生成したレジスタに対応する定数フラグ(Const-FLAG)がセットされている場合には、履歴マスク(P-Mask)のうちロード対象となったバイト位置がセットされる。

[0495] 以上の定数フラグ(Const-FLAG)、変更フラグ(C-FLAG)、および履歴マスク(P-Mask)の設定の制御は、命令区間記憶部2に設けられているRB登録処理部2Aによって行われる。このRB登録処理部2Aは、小さなプロセッサによって構成され、上記のような判断を行うことによって定数フラグ(Const-FLAG)、変更フラグ(C-FLAG)、および履歴マスク(P-Mask)の設定を行う。

[0496] (命令区間例)

ここで、命令区間の一例として、図36(a)に示す命令区間が実行された場合の例について説明する。同図において、PCは、該命令区間が開始された際のPC値を示している。すなわち、命令区間の先頭が1000番地となっている。この命令区間は、ループ構造となっており、11個の命令から構成されている。また、図36(b)は、上記命令区間が実行された場合に、RBに登録される入力アドレスおよび入力データ、並びに出力アドレスおよび出力データを簡略化して示している。

[0497] 第1行目の命令(以降、単に第1の命令のように称する)において、アドレス定数A1がレジスタR1にセットされる。第2の命令において、レジスタR1の内容を用いて、アド

レスA1の内容(00010004)がレジスタRxにロードされる。

[0498] 第3の命令において、アドレス定数A2がレジスタR2にセットされる。第4の命令において、レジスタR2の内容を用いて、アドレスA2の内容(80000000)がレジスタRyにロードされる。

[0499] 第5の命令において、レジスタRxの内容から4を減じた値をアドレスとするアドレスA3(00010000)の内容(0000AAAA)がレジスタRzにロードされる。第6の命令において、レジスタRxの内容に4を加えた値(00010008)がレジスタRxにセットされる。

[0500] 第7の命令において、レジスタRxの内容(00010008)が、レジスタR1の内容を用いてアドレスA1にストアされる。第8の命令において、レジスタRyの内容(80000000)を右に1ビットシフトした値(40000000)がレジスタRyにセットされる。

[0501] 第9の命令において、レジスタRyの内容(40000000)が、レジスタR2を用いてアドレスA2にストアされる。第10の命令において、レジスタRyの内容とレジスタRzの内容とを加えた値(4000AAAA)が、レジスタRzにセットされる。

[0502] 第11の命令において、レジスタRzの内容(4000AAAA)がレジスタRxを用いてアドレスA4にストアされる。第12の命令において、ループの先頭アドレスとしての1000番地に処理が分岐される。

[0503] 上記の第12の命令に引き続いて行われる第2回目のループ処理の例を図36(c)に示し、図36(d)に、この場合のRBに登録される入力アドレスおよび入力データ、並びに出力アドレスおよび出力データを簡略化して示す。また、第2回目のループ処理に引き続いて行われる第3回目のループ処理の例を図36(e)に示し、図36(f)に、この場合のRBに登録される入力アドレスおよび入力データ、並びに出力アドレスおよび出力データを簡略化して示す。

[0504] 以上のように、ループ第1回目では、アドレスA1の値(00010004)、アドレスA2の値(80000000)、アドレス(00010000)の値(0000AAAA)が入力となっており、レジスタRxの値(00010008)、レジスタRyの値(40000000)、レジスタRzの値(4000AAAA)、アドレスA1の値(00010008)、アドレスA2の値(40000000)、アドレス(00010004)の値(4000AAAA)が出力となっている。

[0505] また、ループ第2回目では、アドレスA1の値(00010008)、アドレスA2の値(

40000000)、アドレス(00010004)の値(4000AAAA)が入力となっており、レジスタRxの値(0001000C)、レジスタRyの値(20000000)、レジスタRzの値(6000AAAA)、アドレスA1の値(0001000C)、アドレスA2の値(20000000)、アドレス(00010008)の値(6000AAAA)が出力となっている。

[0506] 以上の処理において、注目すべき点は、ループ第1回目とループ第2回目との間におけるデータの依存関係である。第1の依存関係は、定数アドレスA1に関するループ第1回目の第7の命令とループ第2回目の第2の命令との依存関係である。この依存関係において、定数アドレスA1の値の変化量は増分4と一定である。

[0507] 第2の依存関係は、定数アドレスA2に関するループ第1回目の第9の命令とループ第2回目の第4の命令との依存関係である。この依存関係において、定数アドレスA2の値は右1ビットシフトのため変化量は不定である。

[0508] 第3の依存関係は、変化するアドレスA4に関するループ第1回目の第11の命令とループ第2回目の第5の命令との依存関係である。この依存関係において、アドレスA4のアドレスの変化量は増分4と一定、また、値の変化量は不定である。

[0509] このようなループ構造をループ間の並列処理によって高速化するためには、データ依存関係を動的に把握し、依存関係にない部分を効率よく並列化することが必要である。

[0510] (参考例による命令区間の実行例)

次に、図36(a)に示す命令区間が、上記した参考例におけるRFおよびRBの構成によって実行された場合の例について説明する。図37は、図36(a)に示す命令区間が実行された場合のRBにおける実際の登録状況を示している。

[0511] 第1の命令において、アドレス定数A1がレジスタR1にセットされる。この命令は、定数をセットする命令であるので、レジスタR1に対応する定数フラグ(Const-FLAG)がセットされる。

[0512] 第2の命令において、レジスタR1の内容を用いて、アドレスA1の内容(00010004)がレジスタRxにロードされる。この場合、アドレスA1、マスク(FFFFFFFF)、データ(00010004)は、入力としてRBにおけるInput側の第1列に登録され、レジスタ番号Rx、マスク(FFFFFFFF)、およびデータ(00010004)は出力としてRBにおけるOutput側

の第1列に登録される。なお、この時点でレジスタ番号Rxの出力として登録される値は、後の処理で書き換えられるので、図37に示す値とは異なっている。

[0513] また、アドレスとして用いたレジスタR1に対応する定数フラグ(Const-FLAG)がセットされているので、アドレスA1に対応する履歴マスク(P-Mask)がセットされる。ここで、対象となるデータは(00110000)の4バイトデータであるので、これに対応して、アドレスA1に対応する履歴マスク(P-Mask)には(FFFFFFFF)がセットされる。そして、レジスタRxは、定数がセットされるものではないことになるので、レジスタRxに対応する定数フラグ(Const-FLAG)はリセットされる。

[0514] 第3の命令において、アドレス定数A2がレジスタR2にセットされる。この命令は、定数をセットする命令であるので、レジスタR2に対応する定数フラグ(Const-FLAG)がセットされる。

[0515] 第4の命令において、レジスタR2の内容を用いて、アドレスA2の内容(80000000)がレジスタRyにロードされる。この場合、アドレスA2、マスク(FFFFFFFF)、データ(80000000)は、入力としてRBにおけるInput側の第2列に登録され、レジスタ番号Ry、マスク(FFFFFFFF)、およびデータ(80000000)は出力としてRBにおけるOutput側の第2列に登録される。なお、この時点でレジスタ番号Ryの出力として登録される値は、後の処理で書き換えられるので、図37に示す値とは異なっている。

[0516] また、アドレスとして用いたレジスタR2に対応する定数フラグ(Const-FLAG)がセットされているので、アドレスA2に対応する履歴マスク(P-Mask)がセットされる。ここで、対象となるデータは(80000000)の4バイトデータであるので、これに対応して、アドレスA1に対応する履歴マスク(P-Mask)には(FFFFFFFF)がセットされる。そして、レジスタRyは、定数がセットされるものではないことになるので、レジスタRyに対応する定数フラグ(Const-FLAG)はリセットされる。

[0517] 第5の命令において、レジスタRxの内容から4を減じた値をアドレスとするアドレスA3(00010000)の内容(0000AAAA)がレジスタRzにロードされる。この場合、アドレスA3、マスク(FFFFFFFF)、データ(0000AAAA)は、入力としてRBにおけるInput側の第3列に登録され、レジスタ番号Rz、マスク(FFFFFFFF)、およびデータ(0000AAAA)は出力としてRBにおけるOutput側の第3列に登録される。なお、この時点でレジスタ

番号Rzの出力として登録される値は、後の処理で書き換えられるので、図37に示す値とは異なっている。

[0518] また、アドレスとして用いたレジスタRxに対応する定数フラグ(Const-FLAG)がリセットされているので、アドレスA3に対応する履歴マスク(P-Mask)には(00000000)がセットされる。そして、レジスタRzは、定数がセットされるものではないことになるので、レジスタRzに対応する定数フラグ(Const-FLAG)はリセットされる。

[0519] 第6の命令において、レジスタRxの内容に4を加えた値(00010008)がレジスタRxにセットされる。ここで、レジスタRxはRBにおけるOutput側に既に登録されているので、RBにおけるInput側には登録されない。そして、RBにおけるOutput側に登録されているレジスタRxに対応する値が(00010008)に更新される。

[0520] 第7の命令において、レジスタRxの内容(00010008)が、レジスタR1の内容を用いてアドレスA1にストアされる。ここで、レジスタRxはRBにおけるOutput側に既に登録されているので、RBにおけるInput側には登録されない。アドレスA1、マスク(FFFFFFFF)、およびデータ(00010008)は、出力としてRBにおけるOutput側の第4列に登録される。また、RBにおけるInput側にはアドレスA1が既に登録されているので、アドレスA1に対応する変更フラグ(C-FLAG)がセット(図ではchangeと表示)される。

[0521] 第8の命令において、レジスタRyの内容(80000000)を右に1ビットシフトした値(40000000)がレジスタRyにセットされる。ここで、レジスタRyはRBにおけるOutput側に既に登録されているので、RBにおけるInput側には登録されない。そして、RBにおけるOutput側に登録されているレジスタRyに対応する値が(40000000)に更新される。

[0522] 第9の命令において、レジスタRyの内容(40000000)が、レジスタR2を用いてアドレスA2にストアされる。ここで、レジスタRyはRBにおけるOutput側に既に登録されているので、RBにおけるInput側には登録されない。アドレスA2、マスク(FFFFFFFF)、およびデータ(40000000)は、出力としてRBにおけるOutput側の第5列に登録される。また、RBにおけるInput側にはアドレスA2が既に登録されているので、アドレスA2に対応する変更フラグ(C-FLAG)がセット(図ではchangeと表示)される。

[0523] 第10の命令において、レジスタRyの内容とレジスタRzの内容とを加えた値(4000AAAA)が、レジスタRzにセットされる。ここで、レジスタRyおよびレジスタRzはRBにおけるOutput側に既に登録されているので、RBにおけるInput側には登録されない。そして、RBにおけるOutput側に登録されているレジスタRzに対応する値が(4000 AAAA)に更新される。

[0524] 第11の命令において、レジスタRzの内容(4000AAAA)がレジスタRxを用いてアドレスA4にストアされる。ここで、レジスタRxはRBにおけるOutput側に既に登録されているので、RBにおけるInput側には登録されない。アドレスA4、マスク(FFFFFFFF)、およびデータ(4000AAAA)は、出力としてRBにおけるOutput側の第6列に登録される。

[0525] 第12の命令において、ループの先頭アドレスとしての1000番地に処理が分岐される。後方分岐が検出された時点で、分岐先と登録を開始した命令区間先頭アドレス(1000)とが比較され、一致した場合に、該命令区間の入出力登録が完了する。

[0526] 以上の結果、変更フラグ(C-FLAG)がセットされ、かつ、履歴マスク(P-Mask)がセットされたマスク位置は、アドレスA1、およびアドレスA2となる。このマスク位置に対応するアドレス、マスク、および値が、予測対象として、命令区間ごとに過去の入力履歴を保持する履歴情報として、RBのエントリに記録される。なお、上記の例では出現しなかったが、RBの入力パターンに登録されたレジスタについては無条件に予測対象として履歴として記録される。

[0527] 図38(a)は、図36(a)に示す命令区間が繰り返し実行された場合における、履歴としてRBに登録された例を示している。同図に示すように、RBには、アドレスA1の列に履歴マスク(P-Mask)として(FFFFFFFF)、および、アドレスA2の列に履歴マスク(P-Mask)として(FFFFFFFF)が記憶される。そして、ループの回数が1ー4に変化する間に、各アドレスにおける履歴マスク(P-Mask)に対応する値が変化することになる。各履歴の間に示されるdiffは、対応する入力要素の値の変化量(差分)を示している。このdiffは、予測処理部2Bによって算出される。

[0528] 同図に示す例では、アドレスA1の列に関しては、ループの回数が1ー4に変化する間におけるdiffが全て04となっている。よって、このアドレスに対応する値は、1回のル

ープあたりに04ずつ増加していくことが予想される。一方、アドレスA2の列に関しては、ループの回数が1ー4に変化する間に、diffの値が不定となっている。したがって、アドレスA2に関しては、予測することが困難であることがわかる。

[0529] 以上より、予測処理部2Bは、履歴において、差分が一定となっているアドレスに関して、該差分がその後も継続するものと仮定して予測を行うとともに、差分が一定でない、または差分が0となっているアドレスに関しては予測を行わないようにする。

[0530] 図38(b)は、上記の予測に基づいて、予測処理部2Bがアドレス A1の値に関して予測を行った場合の、予測エントリとしてRBに記録される入力要素の状態を示している。同図において、アドレスA2およびアドレスA7ーA10に関しては、予測値を求めずに直接主記憶3を参照することによって得られたものとなっている。

[0531] このように入力要素の予測値が算出されると、SSP1Bが、この予測入力要素に基づいて命令区間を実行することによって出力要素が算出され、この予測出力要素が予測エントリとしてRBに記憶される。その後、MSP1Aによって命令区間が実行され、予測エントリとしてRBに記憶されている予測入力要素と同じ入力値が入力された場合に、それに対応する予測出力要素を出力することによって再利用が実現されることになる。

[0532] (参考例における課題)

例えばループ制御変数のように、単調変化するアドレス(上記の例ではアドレスA1に対応)の内容については正確に予測することができている。しかしながら、命令区間に配列要素が含まれている場合、配列要素の添字が単調変化していても、配列要素値は一般に単調変化するとは限らない。図36(a)に示す例では、アドレスA1からロードした値が配列要素の添字に該当しており、この添字をアドレスとして用いる主記憶参照(アドレスA3ーA10)はアドレスが変化するために、予測的中率が極めて悪化することになる。ループ間にデータ依存関係がない場合は、キャッシュを直接参照することによって並列処理効果を維持することが可能であるが、例えば図36(a)に示すプログラム例のように、ループ間に依存関係がある場合には、上記したような予測による効果を得ることができない。図39は、参考例による予測に基づいて、ループ処理の2回目および3回目における事前実行を行った結果を示している。同図に示すよう

に、値が確定しないアドレスや、実際の値とは異なる値となっているアドレスが出現しており、予測の効果が薄いことがわかる。

[0533] （予測機構）

RBに対する入出力パターンの登録に関与するアドレスは次のように分類することができる。

(1)第1のタイプのアドレスは、内容が変化しない定数アドレスである。この第1のタイプのアドレスは、内容が変化しないので、再利用の際に過去の値と内容を比較する必要がなく、したがって、内容を予測する必要がないアドレスである。

(2)第2のタイプのアドレスは、内容の変化量が一定となっている定数アドレスである。この第2のタイプのアドレスは、内容の変化量が一定であるので、予測を行うことが可能なアドレスである。上記の例では、アドレスA1が第2のタイプのアドレスに該当する。

(3)第3のタイプのアドレスは、内容の変化量が不定である定数アドレスである。この第3のタイプのアドレスは、予測が困難であるので、書き込みを待ち合わせる必要がある。上記の例では、アドレスA2が第3のタイプのアドレスに該当する。

(4)第4のタイプのアドレスは、アドレス自体は変化するものの、それぞれのアドレスの内容は変化しないアドレスである。すなわち、ストアが発生しないアドレスであり、結果的に内容が変化しないアドレスである。この第4のタイプのアドレスは、内容が変化しないので、再利用の際に過去の値と内容を比較する必要がなく、したがって、内容を予測する必要がないアドレスである。

(5)第5のタイプのアドレスは、アドレス自体が変化し、それぞれのアドレスの内容も、ストアが発生することにより変化するアドレスである。この第5のタイプのアドレスは、内容の変化量が一定であることは期待できず予測は困難であるので、書き込みを待ち合わせる必要がある。上記の例では、アドレスA3ーA10が第5のタイプのアドレスに該当する。

[0534] 本実施形態に係る予測機構は、命令区間の実行時に、上記の第1および第4のタイプのアドレスを除外し、第2、第3、および第5のタイプのアドレスについて動的に分類を行うことを可能としている。また、第5のタイプのアドレスに関しては、事前実行を

行う複数のプロセッサ(MSP1A、SSP1B)間でデータの待ち合わせを行うようにしている。これを実現するために、上記した参考例におけるRBに、さらにストアカウンタ(S-Count)という項目が設けられている。図40(a)は、RBにおける入出力記録行の例を示しており、図40(b)は、履歴格納行の例を示している。

[0535] まず、RBにおける、MSP1AまたはSSP1Bによる命令区間の実行中の入出力パターンを記録する行としての入出力記録行において、出力要素としてのアドレス、すなわちWriteアドレスにストアカウンタ(S-Count)が設けられている。なお、入出力記録行は、MSP1AおよびSSP1Bのそれぞれに対応して設けられている。

[0536] このストアカウンタ(S-Count)は、MSP1AまたはSSP1Bによって該当アドレスに対してストアが行われた回数を示している。すなわち、MSP1AまたはSSP1Bによって該当アドレスに対してストアが1回行われる毎に、RB登録処理部2Aが、該当エントリのストアカウンタ(S-Count)を1増加させる。

[0537] また、RBにおける、各命令区間に対応する履歴エントリを格納する行としての履歴格納行において、Writeアドレスにストアカウンタ(S-Count)が設けられている。後方分岐命令の実行時に、入出力記録行に対する命令区間の入出力登録が完了すると、該入出力記録行に登録された内容が、該命令区間に対応する履歴格納行に追加される。この際に、入出力記録行に登録されている各出力要素のAddress、Mask、およびストアカウンタ(S-Count)が履歴格納行のWrite側に登録される。

[0538] また、RBにおける履歴格納行において、入力要素としてのアドレス、すなわちReadアドレスにもストアカウンタ(S-Count)が設けられている。RBにおける入出力記録行に登録された入力要素のうち、変更フラグ(C-FLAG)がセットされ、かつ、履歴マスク(P-Mask)がセットされた入力要素が、該命令区間に対応する履歴格納行に追加される。この際に、入出力記録行に登録されているAddress、履歴マスク(P-Mask)、およびValueが履歴格納行のRead側に登録される。さらに、RBにおける入出力記録行に登録された入力要素の全てのアドレスのうち、該当命令区間の前回の実行時における入出力パターンが記憶されている履歴格納行のWriteアドレスに含まれているアドレスと一致するアドレスが、該命令区間に対応する履歴格納行に追加される。この際に、入出力記録行に登録されている該当入力要素のAddress、履歴マスク(

P-Mask)、およびストアカウンタ(S-Count)が履歴格納行のRead側に登録される。ここで登録されるストアカウンタ(S-Count)の値は、該当入力要素のアドレスと一致する、前回の命令区間実行時の入出力パターンが記憶されている履歴格納行のWriteアドレスにおけるストアカウンタ(S-Count)値となる。

[0539] (アドレスの分類手法)

以上のような構成のRBによって、上記した第2、第3、および第5のタイプのアドレスをどのように分類するかについて以下に説明する。図41(a)は、図36(a)に示す命令区間が繰り返し実行された場合における、履歴格納行の登録例を示しており、図41(b)は、図41(a)に示す履歴に基づいて、予測処理部2Bが以下に示す予測処理を行った際の、予測値格納領域および待機要アドレス格納領域の例を示している。

[0540] 各命令区間に対応する履歴格納行に登録されている入力要素に履歴マスク(P-Mask)がセットされている場合、予測処理部2Bは、Addressの変化量およびValueの変化量を求める。Addressの変化量が一定の場合には、予測処理部2Bは、今後も変化量が一定であるものとして予測される外挿値を、該当入力要素に対応する予測Addressとして予測値格納領域に格納する。一方、Addressの変化量が不定である場合には、予測処理部2Bは、最後に出現したAddressを該当入力要素の予測Addressとして予測値格納領域に格納する。

[0541] Valueの変化量が一定の場合には、予測処理部2Bは、今後も変化量が一定であるものとして予測される外挿値を、該当入力要素に対応する予測Valueとして設定する。そして、RBにおける予測値格納領域に、該当するAddress、Mask、およびValueを格納する。以上の処理により、上記の第2のタイプのアドレスに関する予測機構が実現される。なお、図41(a)および図41(b)に示す例では、アドレスA1が、Addressの変化量が0で一定、Valueの変化量が04で一定となっており、これに基づいて、第2のタイプのアドレスとして予測値格納領域に登録されている。

[0542] 一方、Valueの変化量が不定の場合には、予測処理部2Bは、RBにおける待機要アドレス格納領域に、該当するAddress、およびMaskを格納するとともに、ストアカウンタ(S-Count)(待機カウンタ)には、予測距離から1を減じた値に、履歴格納行の該当入力要素に対応するストアカウンタ(S-Count)値を乗じた値を格納する。なお、予測

距離とは、該当命令区間が今後繰り返し実行された場合の、現時点からの実行回数を示している。以上のように待機要アドレス格納領域におけるストアカウンタ(S-Count)を設定することによって、待機すべきストアの回数を的確に設定することが可能となる。これにより、上記の第3のタイプのアドレスに関する予測機構が実現される。なお、図41(a)および図41(b)に示す例では、アドレスA2が、履歴マスク(P-Mask)がセットされ、かつ、Valueの変化量が不定となっており、これに基づいて、第3のタイプのアドレスとして待機要アドレス格納領域に登録されている。

[0543] なお、上記した例では、予測処理部2Bは、ストアカウンタ(S-Count)には、予測距離から1を減じた値に、履歴格納行の該当入力要素に対応するストアカウンタ(S-Count)値を乗じた値を格納するようになっているが、次のような処理を行ってもよい。すなわち、予測処理部は、RBにおける予測値格納領域に、該当するAddress、およびMaskを格納するとともに、ストアカウンタ(S-Count)には、履歴格納行の該当入力要素に対応するストアカウンタ(S-Count)値を格納するとともに、予測距離が1だけ短い前回の予測値に基づいて事前実行を開始したSSP1Bを特定する情報を格納してもよい。このようにすれば、全てのSSP1Bによる実行通知のうち、該当するSSP1Bからの実行通知が受信された場合にのみストアカウンタ値を減少させることによって、待機すべきストアの回数を的確に設定することが可能となる。

[0544] 各命令区間に対応する履歴格納行に登録されている入力要素に履歴マスク(P-Mask)がセットされていない場合、予測処理部2Bは、上記と同様に、Addressの変化量およびValueの変化量を求める。Addressの変化量が一定の場合には、予測処理部2Bは、今後も変化量が一定であるものとして予測される外挿値を、該当入力要素に対応する予測Addressとして待機要アドレス格納領域に格納する。一方、Addressの変化量が不定である場合には、予測処理部2Bは、最後に出現したAddressを該当入力要素の予測Addressとして待機要アドレス格納領域に格納する。

[0545] Valueの変化量が一定であることは期待できないので、予測処理部2Bは、RBにおける待機要アドレス格納領域に、該当するAddress、およびMaskを格納するとともに、ストアカウンタ(S-Count)には、履歴格納行の該当入力要素に対応するストアカウンタ(S-Count)値を格納する。なお、この場合には、アドレスが変化しているので、スト

アカウンタ(S-Count)を設定する際に、予測距離を考慮する必要はない。これにより、上記の第5のタイプのアドレスに関する予測機構が実現される。なお、図41(a)および図41(b)に示す例では、アドレスA7ーA10が、履歴マスク(P-Mask)がセットされておらず、かつ、Valueの変化量が不定となっており、これに基づいて、第5のタイプのアドレスとして待機要アドレス格納領域に登録されている。

[0546] (MSP／SSPによる事前実行)

上記のように予測処理部2Bによる処理によって生成された予測値格納行に基づくMSP1A／SSP1Bによる事前実行について説明する。SSP1Bによる事前実行の起動開始後の主記憶読み出しは次のように行われる。

[0547] まずCache／Local7Bが参照されるとともに、以下に示す処理が行われる。

[0548] 最初に、該当SSPに対応する入出力記録行のうち、読み出し対象となる主記憶アドレスと同じアドレスがWrite側に登録されているかをSSP1Bにおける判定部8Bが判定する。登録されている場合には、登録されているValueが読み出し対象となる主記憶アドレスのValueとして読み出される。

[0549] Write側に登録されていない場合には、該当SSPに対応する入出力記録行のうち、読み出し対象となる主記憶アドレスと同じアドレスがRead側のValueに登録されているかをSSP1Bにおける判定部8Bが判定する。登録されている場合には、登録されているValueが読み出し対象となる主記憶アドレスのValueとして読み出される。

[0550] Read側に登録されていない場合には、読み出し対象となる主記憶アドレスと同じアドレスが予測値格納領域に登録されているかをSSP1Bにおける判定部8Bが判定する。登録されている場合には、登録されているValueが読み出し対象となる主記憶アドレスのValueとして読み出される。予測値格納領域に登録されていない場合には、読み出し対象となる主記憶アドレスと同じアドレスが待機要アドレス格納領域に登録されているかをSSP1Bにおける判定部8Bが判定する。登録されている場合、ストアカウンタ(S-Count)値が0より大きい場合には、ストアカウンタ(S-Count)値が0になるまで主記憶読み出しを保留し、Valueに有効な値がセットされた後にValueを参照する。以上のいずれの参照においても、読み出し対象となる主記憶アドレスがなかった場合には、Cache／Local7Bから該当アドレスに関する値の読み込みが行われる。

[0551] また、MSP1A／SSP1Bによる事前実行の起動開始後の主記憶書き込みは次のように行われる。

[0552] MSP1AまたはSSP1Bによってストア命令が実行される場合、その旨が通信部9Aまたは通信部9Bによって、その他の全てのSSP1B…またはMSP1Aに対して通知される。各SSP1Bにおいて、待機要アドレス格納領域の中に、通知されたアドレスと同一のアドレスが登録されている場合、該アドレスのストアカウンタ(S-Count)を1だけ減じてValueに書き込み値を格納する。ただし、ストアカウンタ(S-Count)が既に0である場合には何も行わない。

[0553] 以上のようにしてSSP1Bによって予測事前実行が行われた結果は、RBにおける予測実行結果記憶行に格納される。

[0554] (命令区間の実行例)

上記のようにして予測値が生成された後に、予測値に基づく事前実行を行う場合の例について図42を参照しながら以下に説明する。ここで、予測値は、ループ処理が4回繰り返された結果に基づいて生成されたものとする。また、この例では、SSP1Bを2台利用して実行される例を想定している。この2台のSSP1Bを、それぞれSSP＃1、およびSSP＃2と称する。

[0555] はじめに、MSP1Aがループ5回目の実行を開始し、同時にSSP＃1およびSSP＃2が、それぞれループ6回目およびループ7回目の予測値を受け取って実行を開始したとする。SSP＃1は、SSPのための予測値格納領域にアドレスA1および値(00010018)を保持し、待機要アドレス格納領域に、アドレスA2およびストアカウンタ(S-Count)値としての(0001)、ならびに、アドレスA8およびストアカウンタ(S-Count)値としての(0001)を保持している。同様に、SSP＃2は、SSPのための予測値格納領域にアドレスA1および値(0001001C)を保持し、待機要アドレス格納領域に、アドレスA2およびストアカウンタ(S-Count)値としての(0002)、ならびに、アドレスA9およびストアカウンタ(S-Count)値としての(0001)を保持している。

[0556] SSP＃1は、第2の命令において、レジスタR1を用いてアドレスA1の内容をレジスタRxにロードしている。この時、前記した主記憶読み出し手順に従って、SSPのための予測値格納領域からアドレスA1の値(00010018)を得ている。また、第4の命令に

おいて、レジスタR2を用いてアドレスA2の内容をレジスタRyにロードしている。この時、前記した主記憶読み出し手順に従って、待機要アドレス格納領域からアドレスA2のストアカウンタ(S-Count)値が(0001)であることを認識し、待機する。

[0557] SSP#2は、第2の命令において、レジスタR1を用いてアドレスA1の内容をレジスタRxにロードしている。この時、前記した主記憶読み出し手順に従って、SSPのための予測値格納領域からアドレスA1の値(0001001C)を得ている。また、第4の命令において、レジスタR2を用いてアドレスA2の内容をレジスタRyにロードしている。この時、前記した主記憶読み出し手順に従って、待機要アドレス格納領域から、アドレスA2のストアカウンタ(S-Count)値が(0001)であることを認識し、待機する。

[0558] その後、MSP1Aが第9の命令を実行し、アドレスA2およびストア値(04000000)をSSP#1、およびSSP#2に通知する。SSP#1では、待機要アドレス格納領域のうち、アドレスA2のストアカウンタ(S-Count)値が1だけ減じられることによって0となり、ストア値(04000000)がValueに格納される。これにより、待機状態が終了して第4の命令の実行が完了する。SSP#2では、待機要アドレス格納領域のうち、アドレスA2のストアカウンタ(S-Count)値が1だけ減じられることによって1となり、ストア値(04000000)がValueに格納されるものの、待機状態は継続する。

[0559] SSP#1は、第5の命令において、レジスタRxを用いてアドレスA8の内容をレジスタRxにロードしている。この時、前記した主記憶読み出し手順に従って、待機要アドレス格納領域から、アドレスA8のストアカウンタ(S-Count)値が(0001)であることを認識し、待機する。

[0560] その後、MSP1Aが第11の命令を実行し、アドレスA8およびストア値(7C00AAAA)をSSP#1、およびSSP#2に通知する。SSP#1では、待機要アドレス格納領域のうち、アドレスA8のストアカウンタ(S-Count)値が1だけ減じられることによって0となり、ストア値(7C00AAAA)がValueに格納される。これにより、待機状態が終了して第5の命令の実行が完了する。SSP#2では、待機要アドレス格納領域に該当アドレスがないため、何も実行されず、待機状態が継続する。

[0561] その後、SSP#1が第9の命令を実行し、通知部9Bが、アドレスA2およびストア値(02000000)を全てのSSP1B(SSP#2)に通知する。SSP#2では、待機要アドレス

格納領域のうち、アドレスA2のストアカウンタ(S-Count)値が1だけ減じられることによって0となり、ストア値(02000000)がValueに格納される。これにより、待機状態が終了して第4の命令の実行が完了する。

[0562] さらに、SSP#1が第11の命令を実行し、通知部9Bが、アドレスA9およびストア値(7E00AAAA)を全てのSSP1B(SSP#2)に通知する。SSP#2では、待機要アドレス格納領域のうち、アドレスA9のストアカウンタ(S-Count)値が1だけ減じられることによって0となり、ストア値(7E00AAAA)がValueに格納される。これにより、待機状態が終了して第5の命令の実行が完了する。

[0563] (RF/RBの第2の構成例)

次に、命令区間記憶部2の第2の構成例について、図43を参照しながら以下に説明する。同図に示すように、命令区間記憶部2は、RB、RA、RO1(第2出力パターン記憶手段)、およびRO2(第1出力パターン記憶手段)を備えた構成となっている。

[0564] RBは、比較すべき値であるレジスタ値または主記憶入力値を格納するValue(値格納領域)、およびキー番号を格納するKey(キー格納領域)を備えており、ValueおよびKeyの組み合わせのラインを複数備えている。

[0565] RAは、次に比較すべきレジスタ番号または主記憶アドレスがないことを示す終端フラグE、次に比較すべきレジスタ番号または主記憶アドレスの内容が更新されたことを示す比較要フラグ、次に比較すべき対象がレジスタか主記憶かを示すR/M、次に比較すべきレジスタ番号または主記憶アドレスを示すAdr.(検索項目指定領域)、直前に参照したライン番号を示すUP(親ノード格納領域)、次に比較すべきレジスタ番号または主記憶アドレスよりも優先して比較すべきレジスタ番号または主記憶アドレスを示すAlt.(比較要項目指定領域)、および、優先して比較する際に必要なキーを示すDN(比較要キー指定領域)を備えており、これらはRBにおける各ラインに対応して設けられている。

[0566] RO1およびRO2は、RBおよびRAによる検索結果により、再利用が可能であると判定された場合に、主記憶および/またはレジスタに出力する出力値を格納するものである。RO1は、RAの各ラインに1対1で対応して出力値および出力すべきアドレスを格納している。RO2は、RO1のみでは出力値を格納しきれない場合に、格納し

きれない分の出力値および出力すべきアドレスを格納している。RO2からも出力値を読み出す必要がある場合には、RO1における該当ラインに、RO2における出力値が格納されているポインタが示されており、このポインタを用いてRO2から出力値の読み出しが行われる。また、RBおよびRAは、それぞれCAMおよびRAMによって構成されている。

[0567] （第2の構成例における連想検索動作）

次に、第2の構成例における連想検索動作について説明する。図34に示した構成では、RBにおける各エントリとしての横の行は、一致比較を行うべき入力値の項目を全て含んだものとなっている。すなわち、全ての入力パターンをそれぞれ1つの行としてRBに登録するようになっている。

[0568] これに対して、第2の構成例では、一致比較を行うべき入力値の項目を短い単位に区切り、それぞれの比較単位をノードとしてとらえ、入力パターンを木構造として、アドレス管理表としてのRA、およびRBに登録するようになっている。そして、再利用を行う際には、一致するノードを順次選択することによって、最終的に再利用可能かを判断するようになっている。別の言い方をすれば、複数の入力パターンに共通する部分を1つにまとめて、RAおよびRBの1行に対応づけるようになっている。

[0569] これにより、冗長性をなくし、命令区間記憶部2を構成するメモリの利用効率を向上させることが可能となる。また、入力パターンを木構造としているので、1つの入力パターンをRBにおける1つの行としてのエントリに対応付ける必要がないことになる。よって、一致比較を行うべき入力値の項目の数を可変にすることが可能となっている。

[0570] また、RAおよびRBは、入力パターンを木構造として登録しているので、一致比較を行う際には、マルチマッチが行われないことになる。つまり、命令区間記憶部2としては、シングルマッチ機構を有する連想検索メモリであれば実現可能となる。ここで、シングルマッチ機構のみを有する連想検索メモリは一般的に市販されている一方、マルチマッチをシングルマッチと同一性能によって報告可能な連想検索メモリは一般的には市販されていない。すなわち、第2の構成例によれば、市販の連想検索メモリを利用することができるので、より短期間かつ低コストで、本実施形態に係るデータ処理装置を実現することが可能となる。

[0571] 次に、図44を参照しながら、命令区間記憶部2における連想検索動作の具体例について説明する。まず、命令区間の実行が検出されると、プログラムカウンタ(PC)およびレジスタの内容(Reg.)がRBに入力される。そして、RBにおいて、連想検索により、入力されたこれらの値と、RBのValueの列に登録されている命令区間先頭アドレスおよびレジスタ値とが比較され、値が一致する唯一の行(ライン)が候補(マッチライン)として選択される。この例では、RBにおける「01」のラインがマッチラインとして選択される。

[0572] 次に、マッチラインとして選択されたラインのRBにおける番地である「01」が、エンコード結果としてRAに伝達され、キー01に対応するRAにおけるラインが参照される。キー01に対応するRAにおけるラインでは、比較要フラグが「0」であり、比較すべき主記憶アドレスがA1となっている。すなわち、主記憶アドレスA1に関しては、一致比較を行う必要はないことになる。

[0573] 次に、キー01を用いて、RBにおけるKeyの列に対して検索が行われる。この例では、RBにおける「03」のラインがマッチラインとして選択される。そして、エンコード結果としてキー03がRAに伝達され、キー03に対応するRAにおけるラインが参照される。キー03に対応するRAにおけるラインでは、比較要フラグが「1」であり、比較すべき主記憶アドレスがA2となっている。すなわち、主記憶アドレスA2に関しては、一致比較を行う必要があることになる。ここで、主記憶3における主記憶アドレスA2の値がCache7Aを介して読み出され、RBにおいて、Valueが主記憶3から読み出された値であり、かつ、Keyが「03」となっているラインが検索される。図44に示す例では、Keyが「03」となっているラインは「04」および「05」の2つあるが、主記憶3から読み出された値が「00」であるので、「05」のラインがマッチラインとして選択され、RAに対して、エンコード結果としてキー05が伝達される。

[0574] 以上のような処理が繰り返され、RAにおいて、次に比較すべきレジスタ番号または主記憶アドレスがないことを示す終端フラグEが検出された場合、入力パターンが全て一致したと判定され、該当命令区間は再利用可能と判断される。そして、終端フラグEが検出されたラインから「Select Output」信号が出力され、RO1およびRO2に格納されている、該ラインに対応する出力値がレジスタ6Aおよび主記憶3に対して出力

される。

[0575] 以上のように、第2の構成例による連想検索動作は、次のような特徴を有している。まず、内容が一致したことを示すマッチラインは、RBにおいて1つのラインのみとなるので、検索動作を次列へ伝搬する際にエンコードした結果を1つ伝送すればよいことになる。したがって、RBとRAとの間を接続する信号線は、アドレスのエンコード結果である1組(N本)でよいことになる。これに対して、上記した図1に示す例では、RBにおいてマルチマッチが許容されているので、RBにおける各列同士を接続する信号線は、各ラインごとに設ける(2N本)必要があることになる。すなわち、第2の構成例によれば、命令区間記憶部2を構成する連想検索メモリにおける信号線の数を大幅に低減することが可能となる。

[0576] また、検索途中ではシングルマッチのみが許容されるようになっているので、比較すべき項目の比較順番は、木構造における参照順に限定されることになる。すなわち、レジスタ値とメモリ内容とは、参照順に混在させながら比較する必要がある。

[0577] 入力パターンは、各項目を参照すべきKeyという形でリンクさせることにより、木構造によってRBおよびRAに登録されている。また、入力パターンの項目は、終端フラグによってその終端が示されるようになっている。よって、入力パターンの項目数を可変とすることができるので、再利用表に登録すべき命令区間の状態に応じて、柔軟に入力パターンの項目数を設定することが可能となる。また、入力パターンの項目数が固定でないことによって、利用しない項目が無駄にメモリ領域を占有することがなくなるので、メモリ領域の利用効率を向上させることができる。

[0578] また、木構造によって入力パターンが登録されるので、項目の内容が重複する部分については、複数の入力パターンで1つのラインを共有することが可能となっている。よって、メモリ領域の利用効率をさらに向上させることができる。

[0579] なお、以上のような構成の場合、RAおよびRBを構成するメモリとしては、構造が縦長のものとなる。例えばこのメモリ容量を2Mbyteとした場合、横が8word、縦を65536ラインとすることになる。

[0580] (連想検索動作の別の例)

上記の例では、図43に示したRAにおいて、UP、Alt.、およびDNの項目は利用して

いないことになる。すなわち、上記の例では、RAにおいて、これらの項目を設ける必要はないことになる。これに対して、UP、Alt.、およびDNの項目を利用することによって、連想検索動作をさらに高速化することが可能である。なお、この連想検索動作の別の例は、実施の形態1における(入力パターンを木構造として登録する第2構成例)において説明した内容と同様となるので、ここではその説明を省略する。なお、本実施形態のRAは、実施の形態1におけるRFに対応する。

[0581] (出力値の格納手段)

入力パターンの一致が確認された後に、再利用として出力される出力値を格納する手段についても、実施の形態1における(出力値の格納手段構成例)において説明した内容と同様となるので、ここではその説明を省略する。

[0582] (命令区間記憶部に対する登録処理)

ある命令区間の実行に際して、再利用が行えないと判断された場合に、該命令区間による入出力をRA、RB、RO1、およびRO2に登録する際の動作についても、実施の形態1における(命令区間記憶部に対する登録処理)において説明した内容と同様となるので、ここではその説明を省略する。

[0583] (第2の構成例における予測機構)

図45は、第2の構成例を適用した場合のデータ処理装置の概略構成を示している。図35に示す構成と異なる点としては、RW4A・4Bに入出力記録行が設けられている点、命令区間記憶部2において、RFに区間毎情報として履歴格納行、予測値格納領域、および待機要アドレス格納領域が設けられている点、および上記した第2の構成例におけるRB、RA、W1が設けられている点である。なお、W1は、上記したRO1・RO2に相当するものである。その他の構成については、図35に示す構成と同様であるので、ここではその説明を省略する。

[0584] 第2の構成例では、命令区間の実行時における入出力パターンを一時的に格納する場所としての入出力記録行は、上記のようにRW4A・4Bとなる。ここで、前記した第1の構成例では、命令区間の実行時における入出力パターンはRBに直接登録されていたので、RW4A・4BはRBの各行に対するポインタによって実現されていた。これに対して、第2の構成例では、RAおよびRBが木構造によって構成されているので

、RW4A・4Bが直接RBの行をポイントすることができない。すなわち、第2の構成例では、RW4A・4Bは、RBの各行に対するポインタとして機能するものではなく、命令区間の実行時における入出力パターンを一時的に格納する実質的なメモリとして機能することになる。

[0585] また、図43においては図示していないが、第2の構成例においても、所定の命令区間が繰り返し実行された場合における入力パターンの履歴エントリ、および予測エントリを格納する一時格納メモリ領域として、図34に示すようなRFおよびRBがRFとして設けられている。ただし、この場合には、RBにおけるエントリの行は、履歴エントリを格納する履歴格納行、予測値格納領域、および待機要アドレス格納領域としての数行によって構成されることになる。

[0586] 命令区間が実行されると、その入力要素がRW4A・4Bに順次格納され、全ての入力要素が揃い、演算が行われることによって出力要素が確定すると、この入出力パターンが、上記履歴格納行に格納されるとともに、上記のような木構造の入出力パターン格納機構に格納されることになる。

[0587] また、所定の命令区間が繰り返し実行された場合には、履歴格納行に順次格納され、所定の数の履歴が格納された時点で、上記のように予測処理部2Bによって予測が行われ、予測に基づいてSSP1Bによって実行された結果は、上記のような木構造の入出力パターン格納機構に格納されることになる。

[0588] (本発明の適用例)

「LIMIT」などによって大域変数領域とスタック領域とを区別できるプログラム実行環境があるとした上で、本発明に係るデータ処理装置を他の命令セットアーキテクチャにも適用するためには、スタックフレーム上の変数が、上位／下位関数のどちらの局所変数であるかを区別する手段が必要である。特に、引数を格納するレジスタが不足し、引数をスタックに格納する場合、呼ばれた関数側ではこの区別をすることができないことになる。

[0589] 本実施の形態で取り上げたSPARCプロセッサでは、引数の先頭6ワードを汎用レジスタに格納しており、6ワード以上の引数を扱う関数は出現頻度が高くないことと、引数がスタックに溢れた時点で再利用ができなくなることの両方を利用することによっ

て、関数／ループの再利用を実現している。SPARCプロセッサ同様に、32本以上の汎用レジスタを有する多くのRISCプロセッサでも、同様の判断をすることによって、本発明のような関数／ループの再利用を実現することが可能である。

産業上の利用の可能性

[0590] 本発明に係るデータ処理装置は、上記したようにSPARCプロセッサに適用することが可能である。また、SPARCプロセッサと同様に、32本以上の汎用レジスタを有する多くのRISCプロセッサにも適用することが可能である。また、このようなプロセッサを備えたゲーム機器、携帯型電話機、および情報家電などに適用することができる。

請求の範囲

[1] 主記憶手段から命令区間を読み出し、演算処理を行った結果を主記憶手段に書き込む処理を行うデータ処理装置において、

上記主記憶手段から読み出した命令区間に基づく演算を行う第1の演算手段と、上記第1の演算手段による上記主記憶手段に対する読み出しおよび書き込み時に用いられるレジスタと、上記第1の演算手段によって命令区間の演算が行われたときの入力パターンおよび出力パターンからなる入出力グループを生成する入出力生成手段と、上記入出力生成手段によって生成された入出力グループを記憶する命令区間記憶手段とを備え、

上記第1の演算手段が、命令区間を実行する際に、該命令区間の入力パターンと、上記命令区間記憶手段に記憶されている入力パターンとが一致した場合、該入力パターンと対応して上記命令区間記憶手段に記憶されている出力パターンをレジスタおよび／または主記憶手段に出力する再利用処理を行い、

上記入出力生成手段が、

出力パターンに含まれる各出力要素が、入力パターンに含まれるどの入力要素を起源とするものであるかを示す依存関係格納部と、

上記依存関係格納部に格納されている情報に基づいて、1以上の上記出力要素を含む出力パターンと、1以上の上記入力要素を含む入力パターンとからなる入出力グループを設定する入出力グループ設定手段とを備えていることを特徴とするデータ処理装置。

[2] 上記入出力グループ設定手段が、ある第1の出力要素の起源となる入力要素の組が、他の第2の出力要素の起源となる入力要素の組に全て含まれている場合に、第2の出力要素の起源となる入力要素の組を入力パターン、第1の出力要素および第2の出力要素を出力パターンとする入出力グループを設定することを特徴とする請求項1記載のデータ処理装置。

[3] 上記入出力グループ設定手段が、ある第1の出力要素の起源となる入力要素の組と、他の第2の出力要素の起源となる入力要素の組との間で、共通の入力要素が存在しない場合に、第1の出力要素の起源となる入力要素の組を入力パターン、第1の

出力要素を出力パターンとする第1の入出力グループ、および、第2の出力要素の起源となる入力要素の組を入力パターン、第2の出力要素を出力パターンとする第2の入出力グループをそれぞれ設定することを特徴とする請求項1記載のデータ処理装置。

[4] 上記依存関係格納部が、上記各出力要素を行成分、上記各入力要素を列成分とする2次元配列メモリによって構成され、該2次元配列メモリの各メモリ要素が、該メモリ要素の行成分に対応する出力要素が、該メモリ要素の列成分に対応する入力要素を起源とするか否かの情報を保持していることを特徴とする請求項1記載のデータ処理装置。

[5] 上記第1の演算手段によって命令区間の演算が行われる際に、レジスタおよび／または主記憶手段から読み出しが行われた場合に、上記入出力生成手段が、

(1)読み出しが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素として依存関係格納部に登録されている場合、該出力要素に対応する依存関係格納部の行成分からなる暫定行列を一時記憶する処理、

(2)読み出しが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素としては依存関係格納部に登録されておらず、入力要素として依存関係格納部に登録されている場合、該入力要素に対応する依存関係格納部の列に対応するメモリ要素を1とし、その他のメモリ要素を0とした暫定行列を一時記憶する処理、および、

(3)読み出しが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素および入力要素のいずれとしても依存関係格納部に登録されていない場合には、該アドレスおよび値を入力要素として依存関係格納部に登録するとともに、該入力要素に対応する依存関係格納部の列に対応するメモリ要素を1とし、その他のメモリ要素を0とした暫定行列を一時記憶する処理を行い、

レジスタおよび／または主記憶手段への書き込みが行われた場合に、上記入出力生成手段が、

(4)書き込みが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素として登録されている場合、登録されている出力要素に対応する出力値を、書き込みが行われた値に更新するとともに、

既に登録されている出力要素に対応する依存関係格納部の行成分を、その時点で一時記憶されている全ての暫定行列の論理和に置き換え、その後、一時記憶されている暫定行列を初期化する処理、および、

(5)書き込みが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素として登録されていない場合、該アドレスおよび値を出力要素として依存関係格納部に登録するとともに、該出力要素に対応する依存関係格納部の行成分を、その時点で一時記憶されている全ての暫定行列の論理和に置き換え、その後、一時記憶されている暫定行列を初期化する処理を行うことを特徴とする請求項4記載のデータ処理装置。

[6] 上記入出力グループ設定手段が、上記2次元配列メモリにおける各行成分間の論理積演算を行う行間論理積比較部を含んでおり、

上記入出力グループ設定手段が、依存関係格納部において、ある第1行成分の反転と、ある第2行成分との論理積が全て0になる行成分の組を抽出し、抽出された行成分の組のうち、入力要素の組を最も多く含む行成分以外の行成分を、入出力グループの対象外として設定することを特徴とする請求項4記載のデータ処理装置。

[7] 上記入出力グループ設定手段が、上記2次元配列メモリにおける各行成分間の論理積演算を行う行間論理積比較部を含んでおり、

上記入出力グループ設定手段が、依存関係格納部において、他のどの行成分に対しても論理積が全て0になる行成分を、それぞれ入出力グループとして設定することを特徴とする請求項4記載のデータ処理装置。

[8] 少なくとも1つの第2の演算手段をさらに備え、

上記第2の演算手段が、上記第1の演算手段によって処理が行われている命令区間に関して、今後入力が予想される予測入力値に基づいて該命令区間の演算を行い、その結果を上記命令区間記憶手段に対して登録することを特徴とする請求項1ー7のいずれか一項に記載のデータ処理装置。

[9] 上記入出力グループ設定手段が、

各出力要素が所属する入出力グループの情報を格納する出力側グループ格納部と、

各入力要素が所属する入出力グループの情報を格納する入力側グループ格納部と、

入出力グループを生成している途中に、上記依存関係格納部に変更があった場合に、変更された出力要素と入力要素との依存関係を格納する一時格納部と、

入出力グループを生成している途中に、上記依存関係格納部に変更があった場合に、変更された入出力グループの情報を格納するグループ一時格納部とを備えていることを特徴とする請求項1記載のデータ処理装置。

[10] 上記入出力グループ設定手段が、入出力グループを生成している途中に、上記出力要素および／または上記入力要素に対して既に割り当てられている入出力グループの情報を格納するグループ管理部をさらに備えていることを特徴とする請求項9記載のデータ処理装置。

[11] 上記依存関係格納部が、上記各出力要素を行成分、上記各入力要素を列成分とする2次元配列メモリによって構成され、該2次元配列メモリの各メモリ要素が、該メモリ要素の行成分に対応する出力要素が、該メモリ要素の列成分に対応する入力要素を起源とするか否かの情報を保持していることを特徴とする請求項9記載のデータ処理装置。

[12] 上記一時格納部が、上記依存関係格納部における複数行のメモリ要素の論理和を格納するものであり、

上記グループ一時格納部が、上記出力側グループ格納部における複数行のメモリ要素の論理和、および／または、上記入力側グループ格納部における複数の入力要素に対応するメモリ要素の論理和を格納するものであることを特徴とする請求項11記載のデータ処理装置。

[13] 上記入出力グループ設定手段が、入出力グループを生成している途中に、条件分岐命令が検出された場合に、該条件分岐命令が依存する入力要素の情報を格納する条件分岐格納部をさらに備えていることを特徴とする請求項9記載のデータ処理装置。

[14] 上記第1の演算手段によって命令区間の演算が行われる際に、レジスタおよび／または主記憶手段から読み出しが行われた場合に、上記入出力生成手段が、

(1)読み出しが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素として依存関係格納部に登録されている場合、該出力要素に対応する依存関係格納部の行成分と、上記一時格納部の各要素との論理和を該一時格納部に格納するとともに、該出力要素に対応する出力側グループ格納部の行成分と、上記グループ一時格納部の各要素との論理和を該グループ一時格納部に格納する処理、

(2)読み出しが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素としては依存関係格納部に登録されておらず、入力要素として依存関係格納部に登録されている場合、該入力要素に対応する依存関係格納部の列に対応するメモリ要素を1とし、その他のメモリ要素を0とした情報を上記一時格納部に格納するとともに、該入力要素に対応する入力側グループ格納部の各要素と、上記グループ一時格納部の各要素との論理和を該グループ一時格納部に格納する処理、および、

(3)読み出しが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素および入力要素のいずれとしても依存関係格納部に登録されていない場合には、該アドレスおよび値を入力要素として依存関係格納部に登録するとともに、該入力要素に対応する依存関係格納部の列に対応するメモリ要素を1とし、その他のメモリ要素を0とした情報を上記一時格納部に格納する処理を行い、

レジスタおよび／または主記憶手段への書き込みが行われた場合に、上記入出力生成手段が、

(4)書き込みが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素として登録されている場合、登録されている出力要素に対応する出力値を、書き込みが行われた値に更新するとともに、

既に登録されている出力要素に対応する依存関係格納部の行成分を、その時点で一時記憶されている上記一時格納部に格納されている情報に置き換えるとともに、上記グループ一時格納部に格納されている情報に基づいて、該出力要素に対応する出力側グループ格納部の情報、および、該出力要素が依存する各入力要素に対応する入力側グループ格納部の情報を更新する処理、および、

(5)書き込みが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素として登録されていない場合、該アドレスおよび値を出力要素として依存関係格納

部に登録するとともに、該出力要素に対応する依存関係格納部の行成分を、その時点で一時記憶されている上記一時格納部に格納されている情報に置き換えるとともに、上記グループ一時格納部に格納されている情報に基づいて、該出力要素に対応する出力側グループ格納部の情報、および、該出力要素が依存する各入力要素に対応する入力側グループ格納部の情報を更新する処理を行うことを特徴とする請求項12記載のデータ処理装置。

[15] 上記命令区間記憶手段が、複数の上記入力パターンを、一致比較すべき項目をノードとみなした木構造として記憶する入力パターン記憶手段を備えていることを特徴とする請求項1または9記載のデータ処理装置。

[16] 上記入力パターン記憶手段が、上記入力パターンにおいて一致比較すべき項目の値と、次に比較すべき項目とを対応させて格納することによって、上記木構造を実現することを特徴とする請求項15記載のデータ処理装置。

[17] 上記入力パターン記憶手段が、連想検索手段と、付加記憶手段とを備え、

上記連想検索手段が、一致比較すべき項目の値を格納する値格納領域と、該項目を識別するキーを格納するキー格納領域とを有する1つ以上の検索対象ラインを備え、

上記付加記憶手段が、上記検索対象ラインに対応した対応ラインごとに、次に連想検索を行うべき項目を格納する検索項目指定領域を有していることを特徴とする請求項16記載のデータ処理装置。

[18] 主記憶手段から命令区間を読み出し、演算処理を行った結果を主記憶手段に書き込む処理を行うデータ処理装置において、

上記主記憶手段から読み出した命令区間に基づく演算を行う第1の演算手段と、上記第1の演算手段による上記主記憶手段に対する読み出しおよび書き込み時に用いられるレジスタと、複数の命令区間の実行結果としての入力パターンおよび出力パターンを記憶する入出力記憶手段とを備え、

上記第1の演算手段が、命令区間を実行する際に、該命令区間の入力パターンと、上記入出力記憶手段に記憶されている入力パターンとが一致した場合、該入力パターンと対応して上記入出力記憶手段に記憶されている出力パターンをレジスタお

よび／または主記憶手段に出力する再利用処理を行うとともに、

上記第1の演算手段による命令区間の実行結果を、上記入出力記憶手段に記憶する際に、入力パターンに含まれる入力要素のうち、予測を行うべき入力要素と予測を行う必要のない入力要素とを区別し、この区別情報を上記入出力記憶手段に登録する登録処理手段と、

上記区別情報に基づいて、上記入出力記憶手段に記憶されている入力要素のうち、予測を行うべき入力要素の値の変化の予測を行う予測処理手段と、

上記予測処理手段によって予測された入力要素に基づいて、該当する命令区間を事前実行する第2の演算手段とをさらに備え、

上記第2の演算手段による命令区間の事前実行結果が上記入出力記憶手段に記憶されることを特徴とするデータ処理装置。

[19] 上記登録処理手段が、入力に用いられた上記レジスタの各アドレスに対して、スタックポインタまたはフレームポインタとして用いられる場合、および、該アドレスに対する書き込み命令が定数セット命令である場合に、該当アドレスに対して区別情報として定数フラグをセットし、上記以外の場合に、該当アドレスに対して上記定数フラグをリセットすることを特徴とする請求項18記載のデータ処理装置。

[20] 上記登録処理手段が、入力要素が新規に上記入出力記憶手段に記憶される際に、該入力要素のアドレスに対して、区別情報として変更フラグをリセットし、上記入出力記憶手段に記憶された後に、該当アドレスに対してストア命令が実行された場合に、該当アドレスに対して変更フラグをセットすることを特徴とする請求項18または19記載のデータ処理装置。

[21] 上記登録処理手段が、入力要素が新規に上記入出力記憶手段に記憶される際に、該入力要素のアドレスに対して、区別情報として履歴フラグをリセットし、該アドレスに対するロード命令実行時に、該アドレスを生成したレジスタアドレスに上記定数フラグがセットされている場合に、該アドレスに対して履歴フラグをセットすることを特徴とする請求項19記載のデータ処理装置。

[22] 上記登録処理手段が、入力要素が新規に上記入出力記憶手段に記憶される際に、該入力要素のアドレスに対して、区別情報として変更フラグをリセットし、上記入出

力記憶手段に記憶された後に、該当アドレスに対してストア命令が実行された場合に、該当アドレスに対して変更フラグをセットするとともに、

上記予測処理手段が、上記入出力記憶手段に記憶されている入力要素のアドレスのうち、上記変更フラグがセットされ、かつ、履歴フラグがセットされているアドレスに関して、入力要素の変化の予測を行うことを特徴とする請求項21記載のデータ処理装置。

[23] 上記予測処理手段が、上記入出力記憶手段に記憶されている入力要素のうち、該入力要素の履歴における値の変化量が0ではない入力要素のみに対して、入力要素の値の変化の予測を行うことを特徴とする請求項18または21記載のデータ処理装置。

[24] 上記登録処理手段が、上記第1の演算手段による命令区間の実行結果を、上記入出力記憶手段に記憶する際に、入力パターンに含まれる入力要素のうち、予測を行うべき入力要素と予測を行う必要のない入力要素とを区別し、この区別情報を上記入出力記憶手段に登録するとともに、上記入出力記憶手段に格納される出力パターンにおける出力要素のうち、該当命令区間の実行の際にストアが行われたものについてそのストアの回数をカウントし、このカウント値を上記入出力記憶手段に格納し、

上記第2の演算手段が、上記予測処理手段によって予測された入力要素に基づいて、該当する命令区間を事前実行するとともに、上記カウント値に基づいて該当入力要素に対して行われるストアの回数を待機した上で主記憶からの読み出しを行って該当する命令区間の事前実行を行うことを特徴とする請求項18記載のデータ処理装置。

[25] 上記入出力記憶手段が、上記第1の演算手段による命令区間の実行結果としての入力パターンおよび出力パターンを一時的に記録する入出力記録領域を備え、

上記入出力記録領域が、各出力要素に対して、ストアが行われた回数を格納するストアカウンタを有することを特徴とする請求項24記載のデータ処理装置。

[26] 上記入出力記憶手段が、上記第1の演算手段によって演算が行われた命令区間毎に過去の実行結果の履歴を格納する履歴格納領域を備え、

上記登録処理手段が、上記入出力記録領域に記録された実行結果を上記履歴格

納領域に格納するとともに、上記入出力記録領域に記録された実行結果の入力パターンに含まれる入力要素のうち、履歴格納領域に前回の実行結果として登録されている出力要素と同じアドレスの入力要素に対して、対応する前回の出力要素のストアカウンタを該入力要素に対するストアカウンタとして登録することを特徴とする請求項25記載のデータ処理装置。

[27] 上記入出力記憶手段が、上記予測処理手段によって予測された入力要素を格納する予測値格納領域を備え、

上記予測処理手段が、上記履歴格納領域に格納されている入力要素のうち、実行履歴の間での値の変化量が一定である入力要素に関して値の予測を行い、上記予測値格納領域に格納することを特徴とする請求項26記載のデータ処理装置。

[28] 上記入出力記憶手段が、ストアの回数を待機した上で主記憶からの読み出しを行うべき入力要素を格納する待機要アドレス格納領域を備え、

上記予測処理手段が、上記履歴格納領域に格納されている入力要素のうち、実行履歴においてアドレスが変化せず、実行履歴の間での値の変化量が不定である入力要素に関して、上記ストアカウンタ、および予測距離に基づく値としての待機カウンタを上記待機要アドレス格納領域に格納することを特徴とする請求項26記載のデータ処理装置。

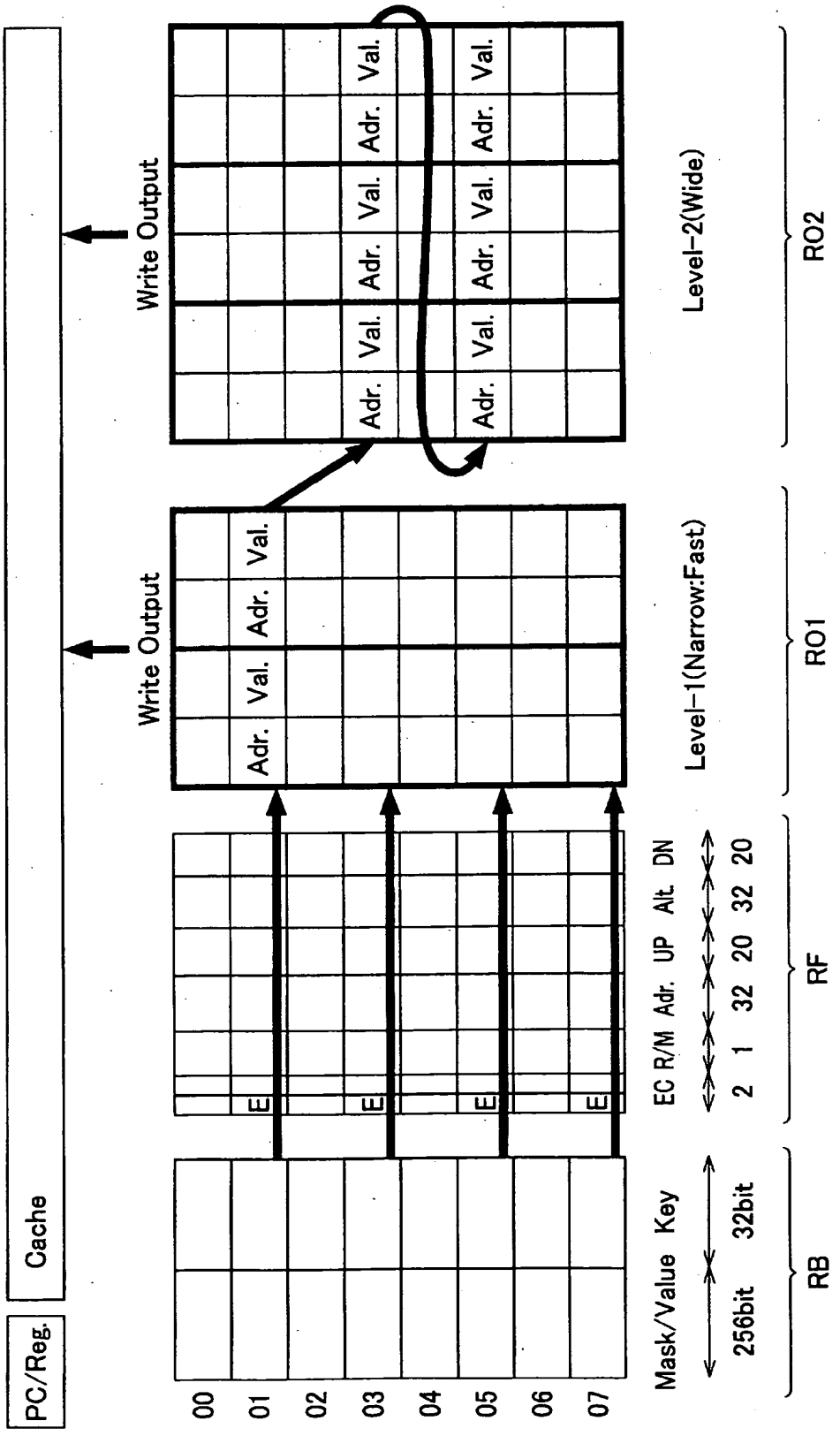
[29] 上記入出力記憶手段が、ストアの回数を待機した上で主記憶からの読み出しを行うべき入力要素を格納する待機要アドレス格納領域を備え、

上記予測処理手段が、上記履歴格納領域に格納されている入力要素のうち、実行履歴においてアドレス自体が変化し、それぞれのアドレスの値も、ストアが発生することにより変化する入力要素に関して、上記ストアカウンタに基づく値としての待機カウンタを上記待機要アドレス格納領域に格納することを特徴とする請求項26記載のデータ処理装置。

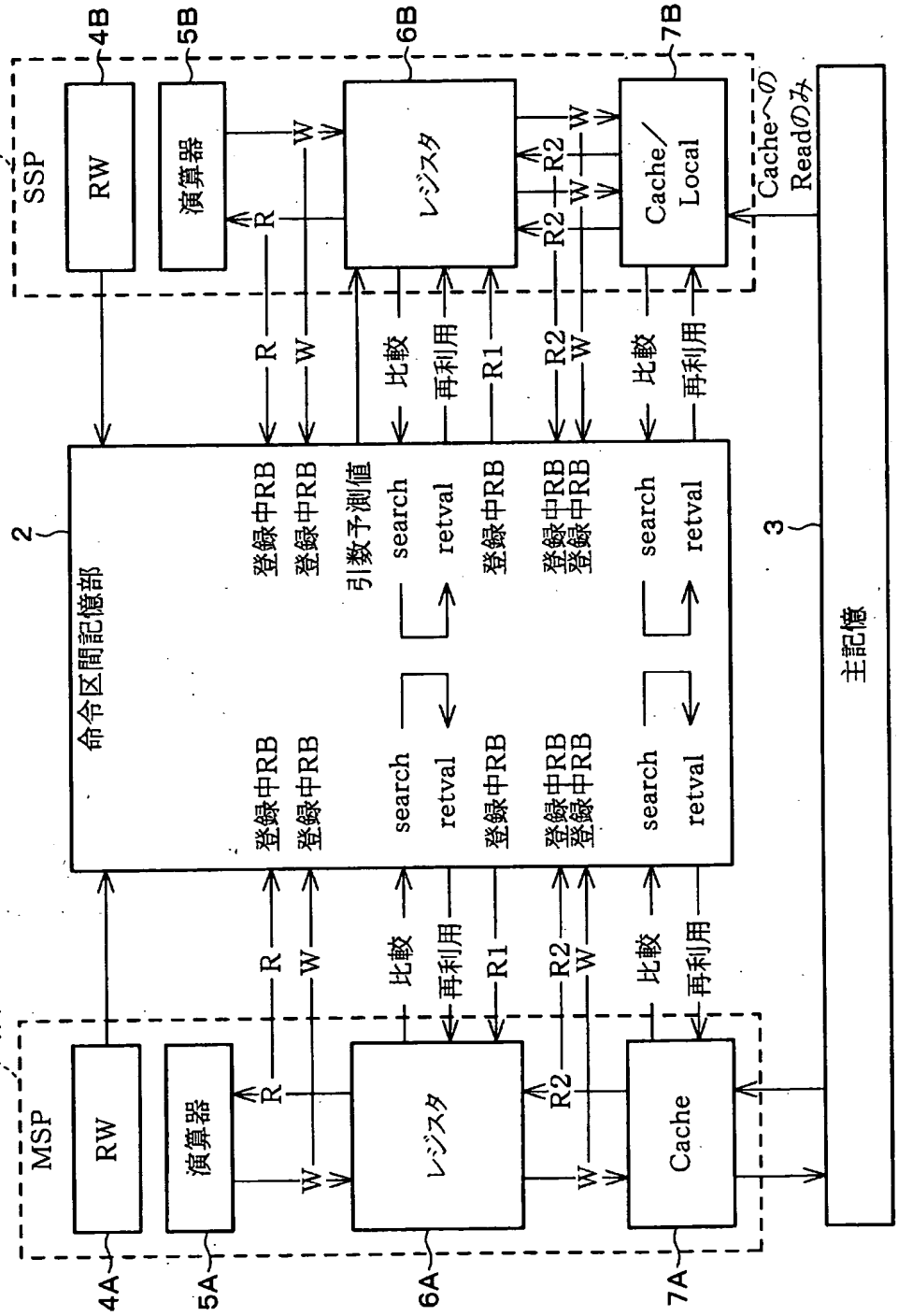
[30] 請求項1ないし29のいずれか一項に記載のデータ処理装置が備える各手段が行う処理をコンピュータに実行させることを特徴とするデータ処理プログラム。

[31] 請求項30に記載のデータ処理プログラムを記録したコンピュータ読取り可能な記録媒体。

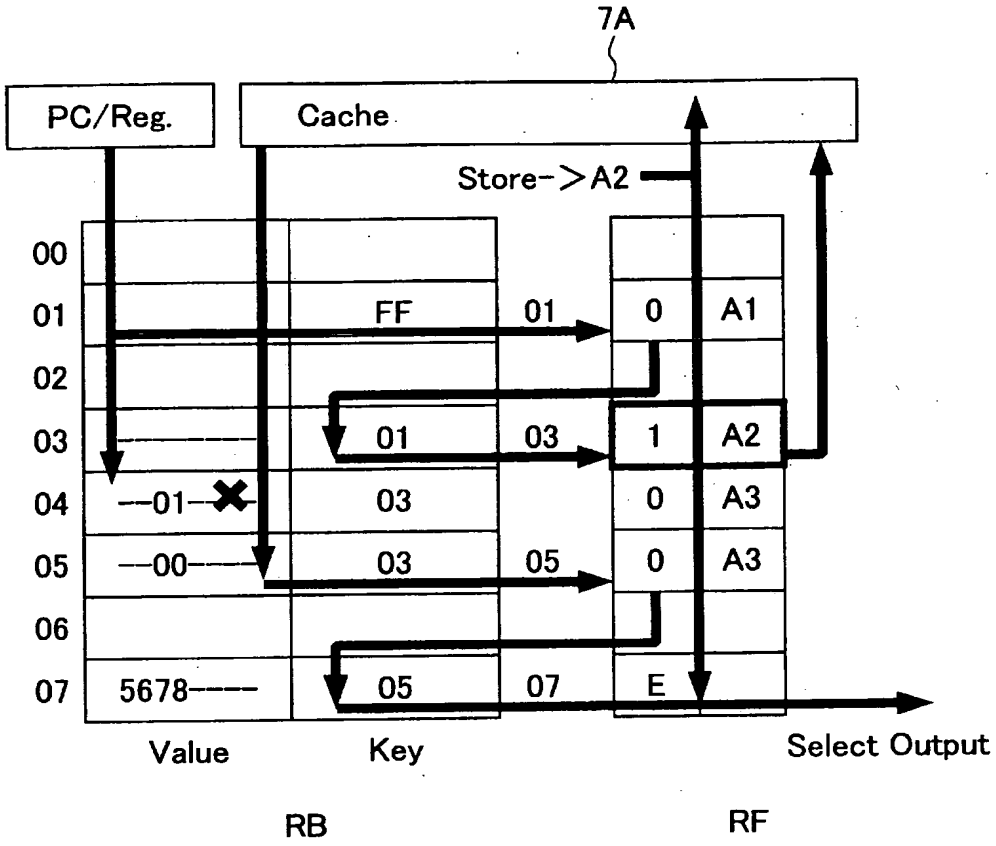
[図1]


PC/Reg.
Cache
Write Output
Write Output
Adr.
Val.
E
00
01
02
03
04
05
06
07
Mask/Value Key
256bit 32bit
EC R/M Adr. UP Alt. DN
2 1 32 20 32 20
Level-1(Narrow:Fast)
Level-2(Wide)
RB
RF
RO1
RO2

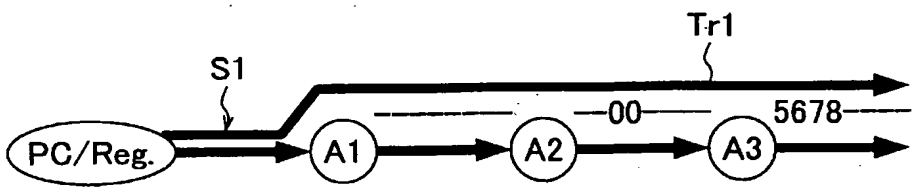
[図2]


1B
SSP
4B
RW
5B
演算器
6B
レジスタ
7B
Cache／Local
Cache～のReadのみ
R
W
R2
比較
再利用
R1
2
命令区間記憶部
登録中RB
引数予測値
search
retval
3
主記憶
1A
MSP
4A
5A
6A
7A
Cache

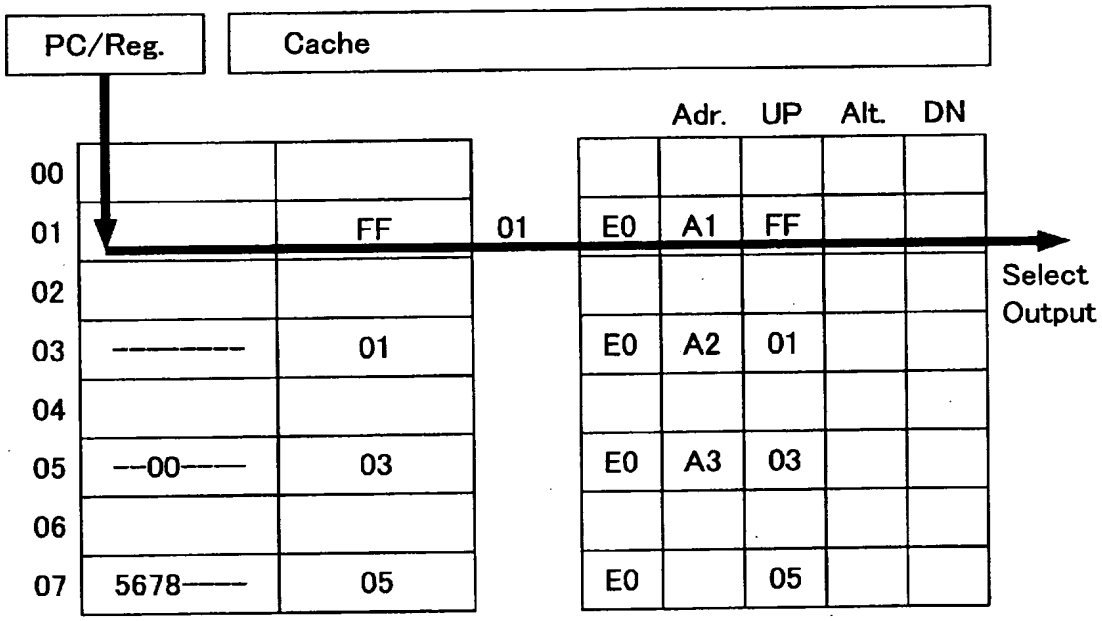
[図3]


7A
PC/Reg.
Cache
Store->A2
00
01
02
03
04
05
06
07
FF
01
03
03
05
07
0
A1
1
A2
0
A3
0
A3
E
—01—
—00—
5678—
Value
Key
Select Output
RB
RF

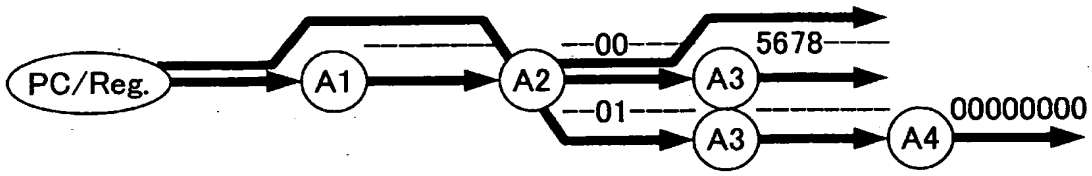
[図4(a)]


Tr1
S1
PC/Reg.
A1
A2
A3
--00--
5678----

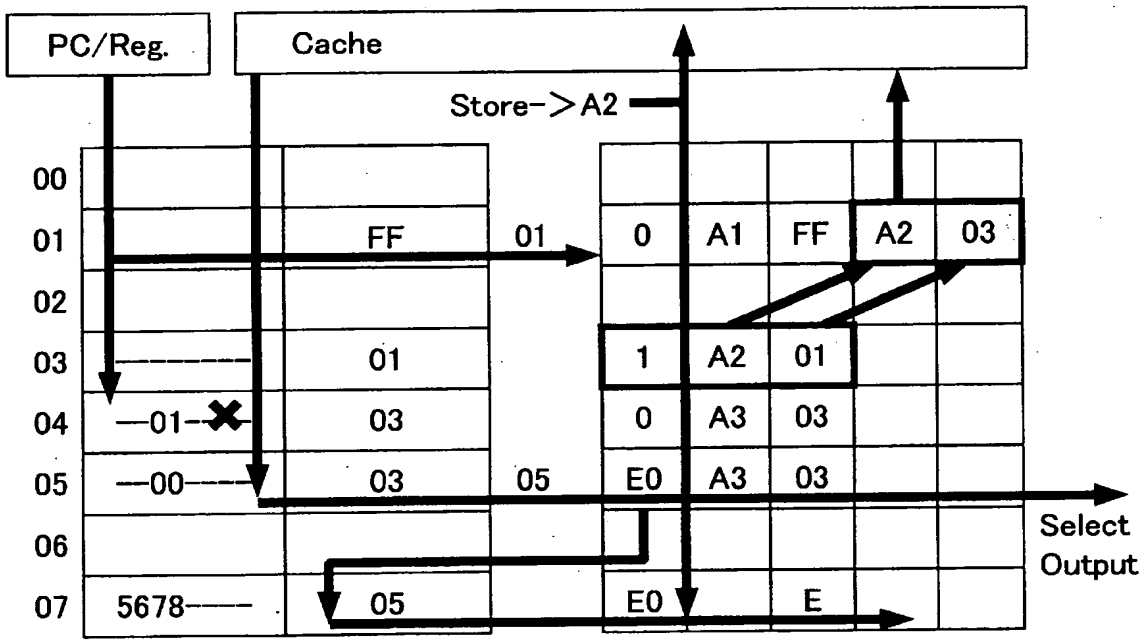
[図4(b)]


PC/Reg.
Cache
Adr.
UP
Alt.
DN
00
01
02
03
04
05
06
07
FF
01
--00--
5678----
01
03
05
E0
A1
FF
A2
01
A3
03
Select
Output

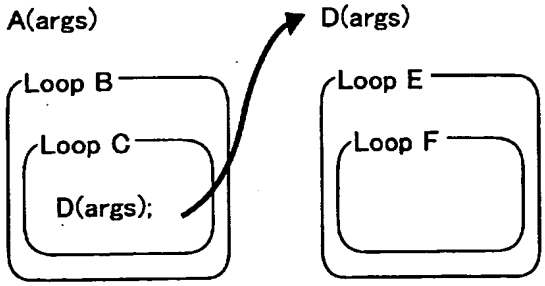
[図5(a)]


PC/Reg.
A1
A2
00
01
A3
A3
5678
00000000
A4

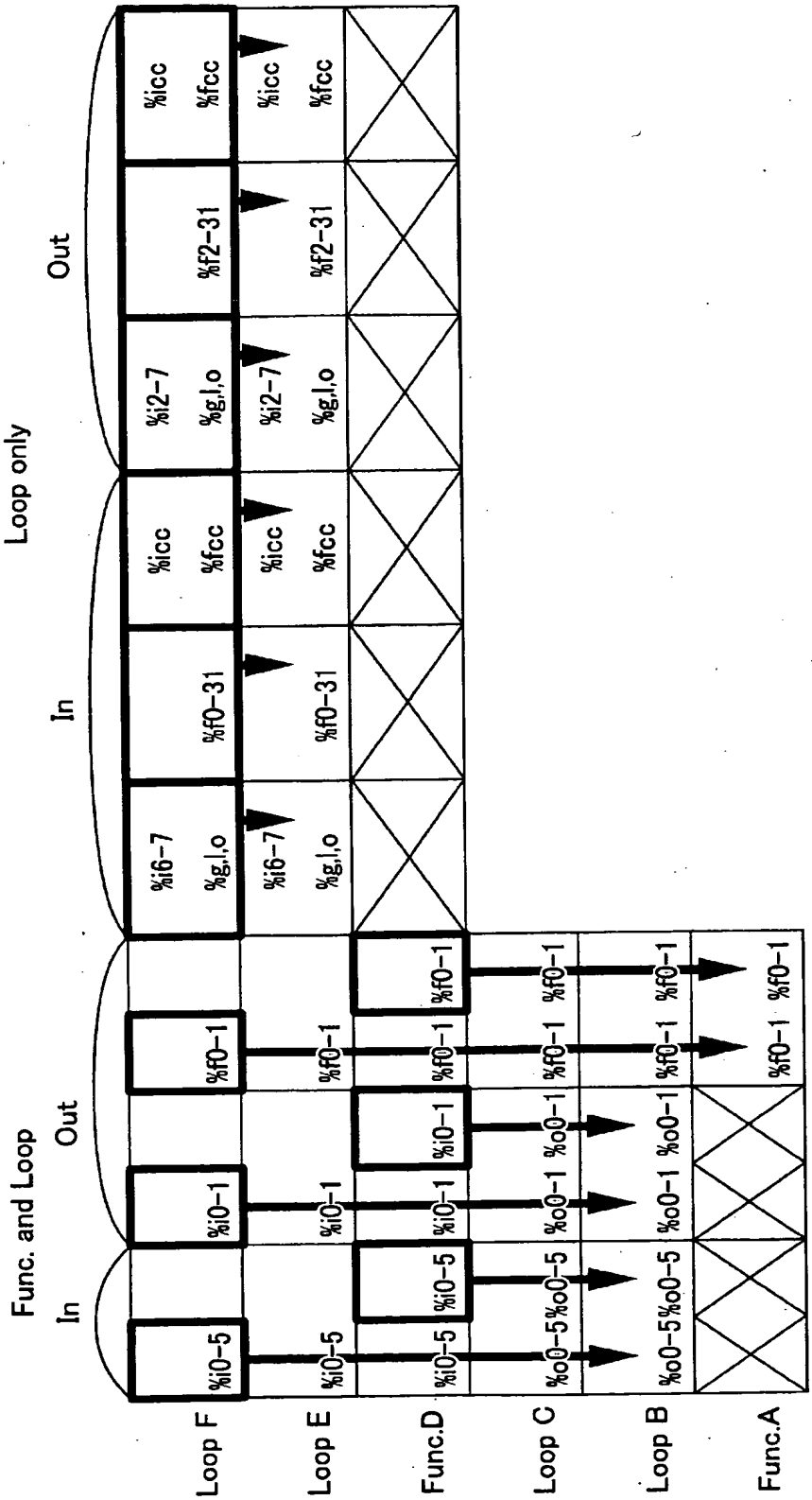
[図5(b)]


PC/Reg.
Cache
Store->A2
00
01
02
03
04
05
06
07
FF
01
01
03
03
5678
05
01
00
05
0
A1
FF
A2
03
1
A2
01
0
A3
03
E0
A3
03
E0
E
Select
Output

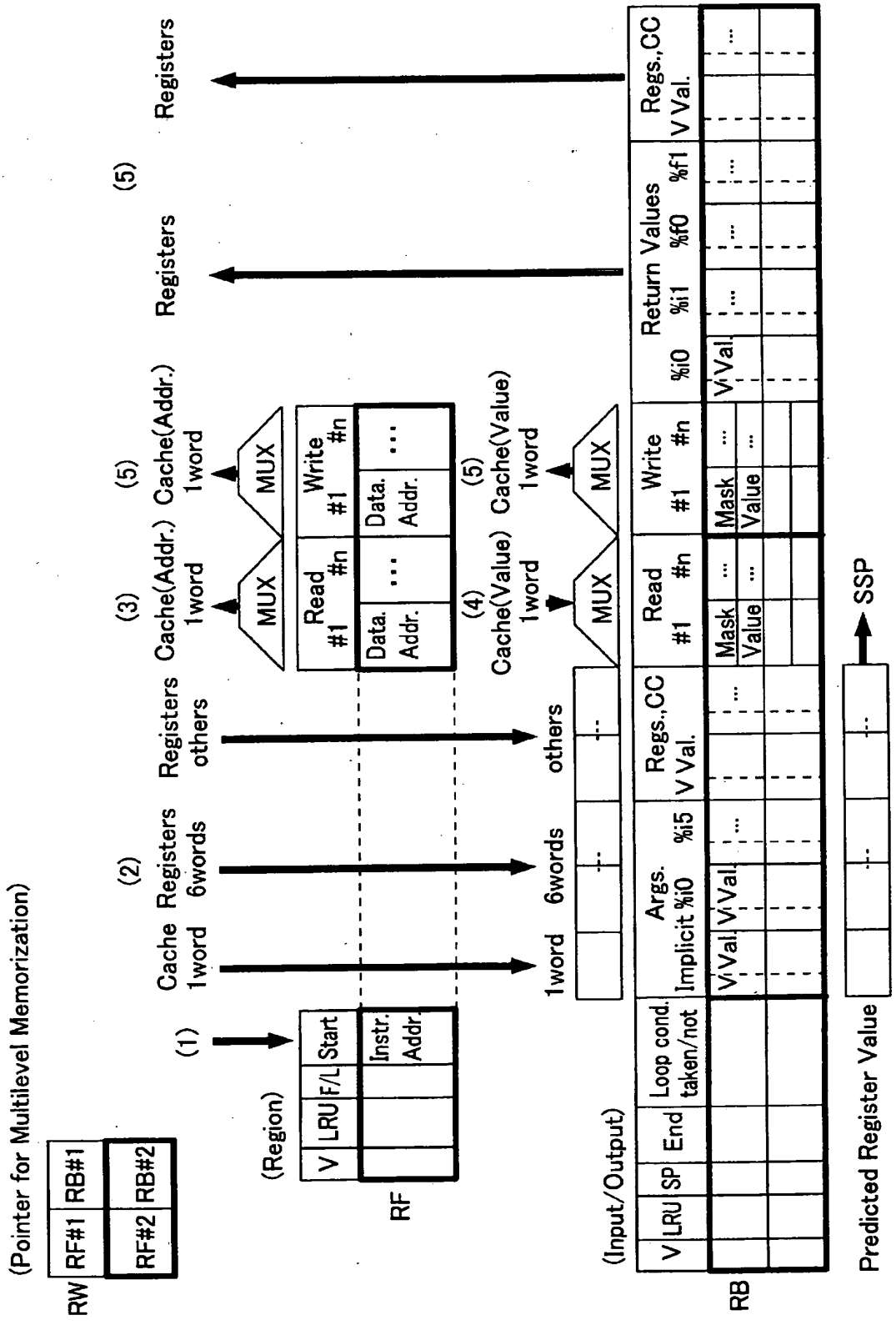
[図6]


A(args)
Loop B
Loop C
D(args);
D(args)
Loop E
Loop F

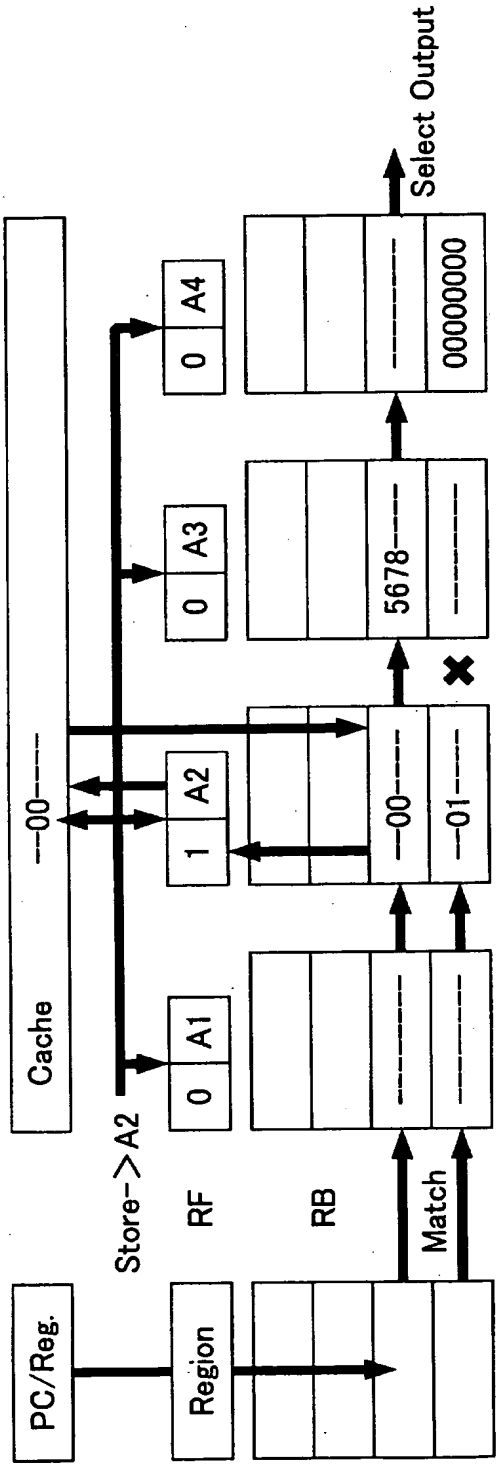
[図7]


Func. and Loop
In
Out
Loop only
In
Out
Loop F
Loop E
Func.D
Loop C
Loop B
Func.A
%i0-5
%i0-1
%f0-1
%i6-7
%g,l,o
%f0-31
%icc
%fcc
%i2-7
%f2-31
%o0-5
%o0-1

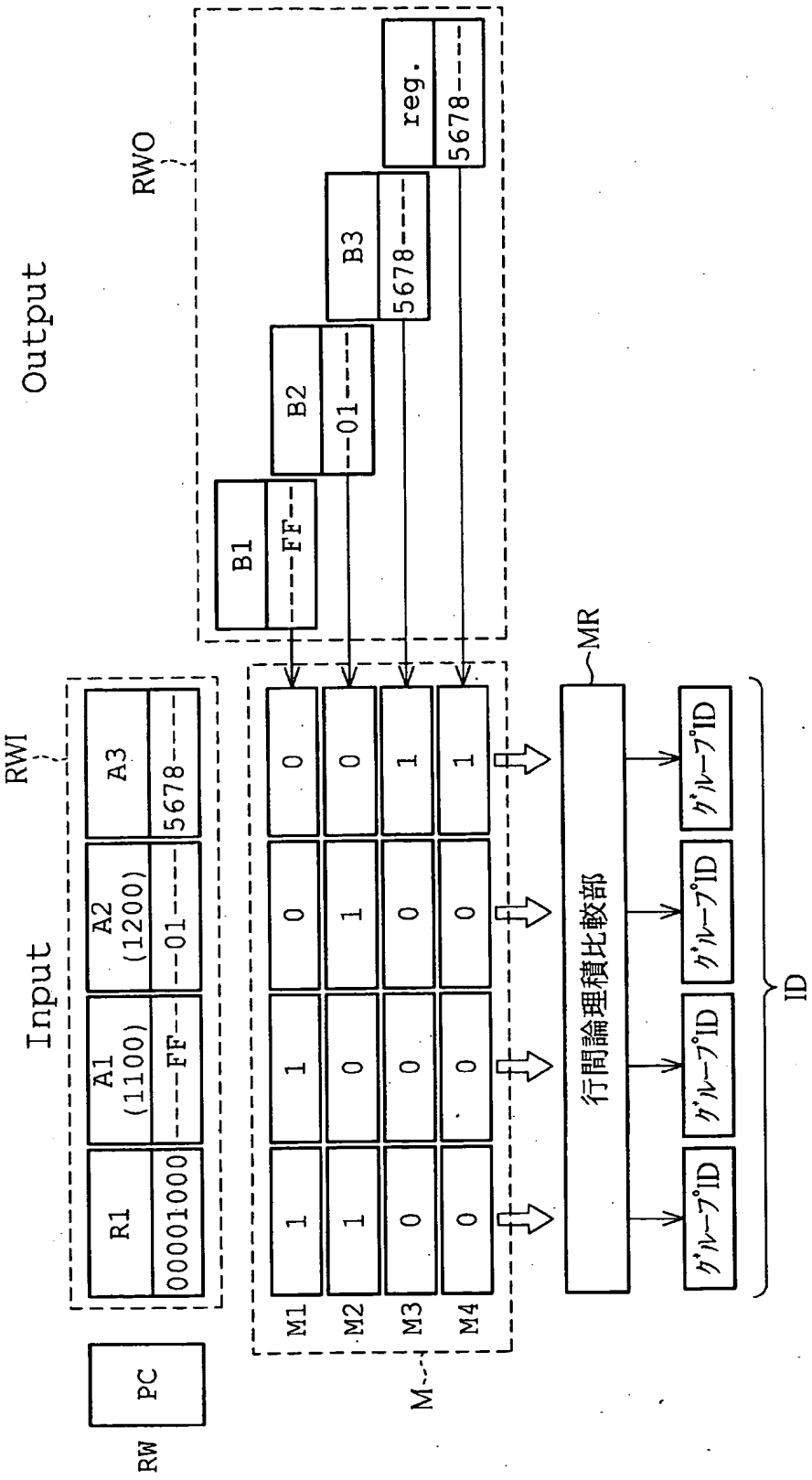
[図8]


(Pointer for Multilevel Memorization)
RW
RF#1
RB#1
RF#2
RB#2
(1)
(Region)
V
LRU
F/L
Start
Instr. Addr
RF
(2)
Cache 1word
Registers 6words
(3)
Cache(Addr.) 1word
MUX
Read #1 #n
Data. Addr.
Write #1 #n
Data. Addr.
(4)
Cache(Value) 1word
(5)
Cache(Addr.) 1word
Cache(Value) 1word
Registers
others
(5)
Registers
(5)
Registers
(Input/Output)
V
LRU
SP
End
Loop cond. taken/not
Implicit %i0
Args. %i5
Regs.,CC V Val.
Read #1 #n
Mask
Value
Write #1 #n
Return Values %i0 %i1 %f0 %f1
Regs.,CC V Val.
RB
Predicted Register Value
SSP

[図9]


Select Output
A4
0
A3
0
A2
1
A1
0
Cache
--00----
Store->A2
RF
RB
Match
PC/Reg.
Region
00000000
5678
--00--
--01--

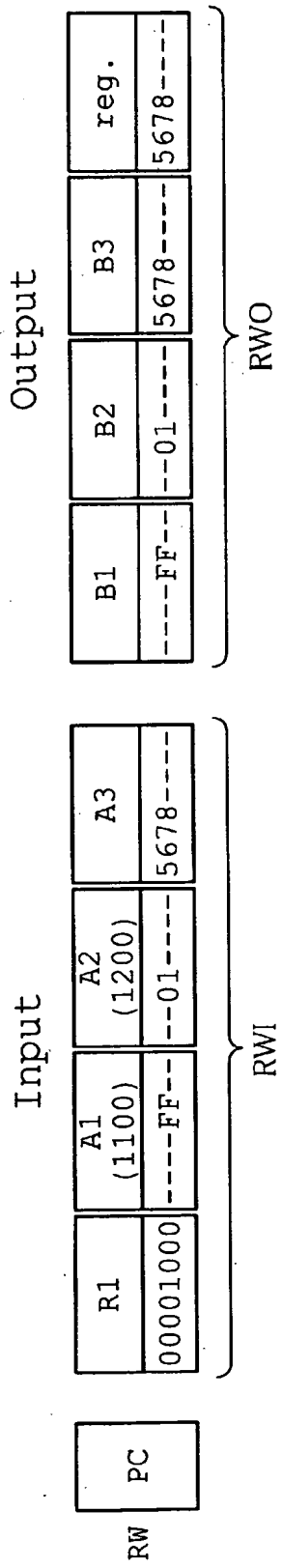
[図10]


RW
PC
RWI
Input
R1
00001000
A1
(1100)
FF
A2
(1200)
01
A3
5678
M
M1
M2
M3
M4
1
1
0
0
1
0
1
0
0
0
1
0
0
0
1
0
MR
行間論理積比較部
RWO
Output
B1
FF
B2
01
B3
5678
reg.
5678
グループID
グループID
グループID
グループID
ID

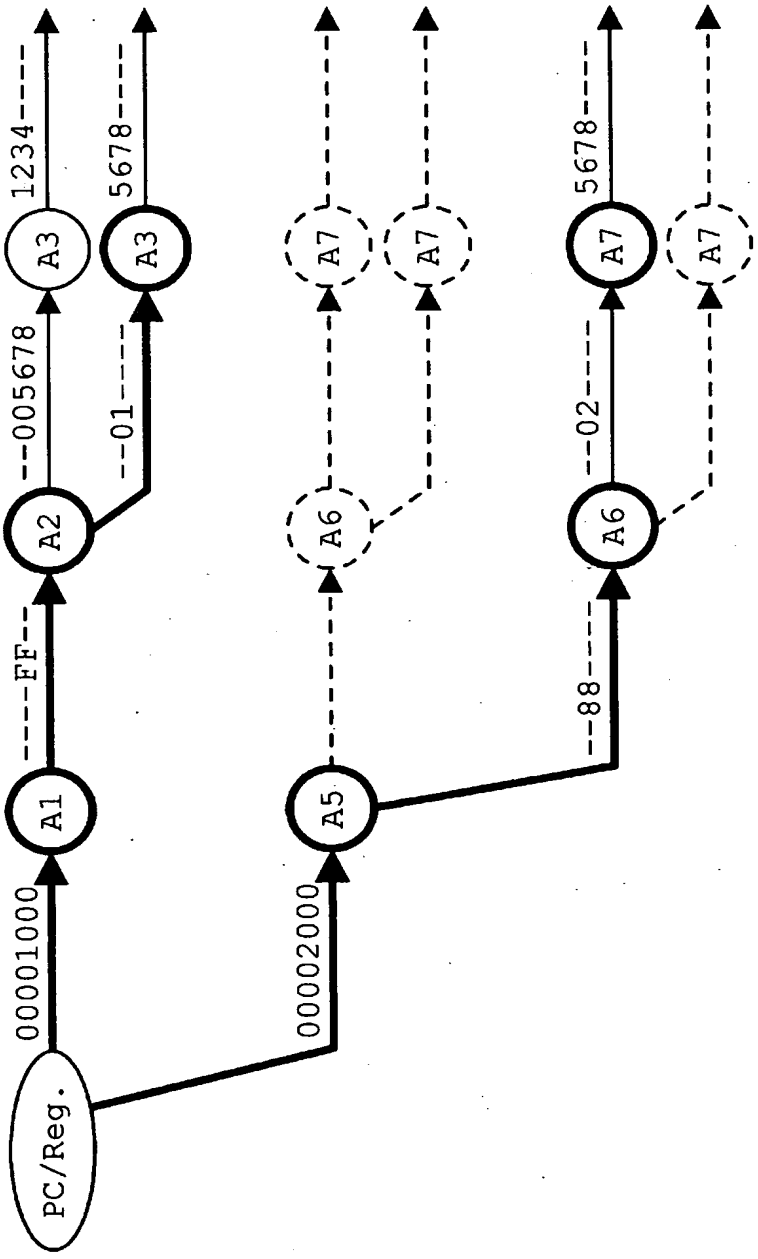
[図11]

```
PC:load  [R1+100](----FF--)   -> reg.
   store reg.                 -> B1(----FF--)
   load  [R1+200](--01----)   -> reg.
   store reg.                 -> B2(--01----)
   load  A3(5678----)         -> reg.
   store reg.                 -> B3(5678----)
```

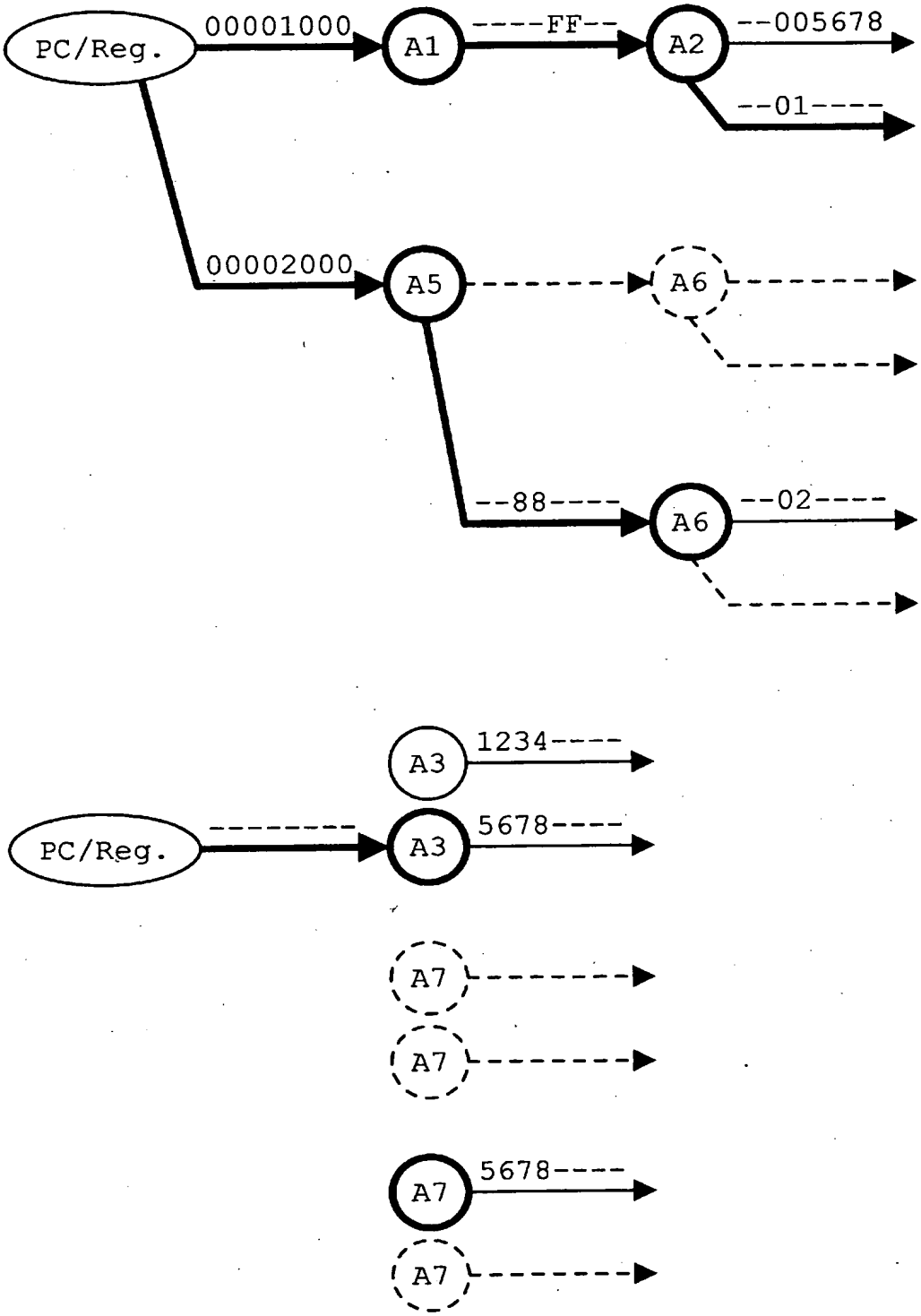
[図12]


Output
B1
B2
B3
reg.
----FF--
--01----
5678----
5678----
RWO
Input
R1
A1 (1100)
A2 (1200)
A3
00001000
----FF--
--01----
5678----
RWI
PC
RW

[図13]

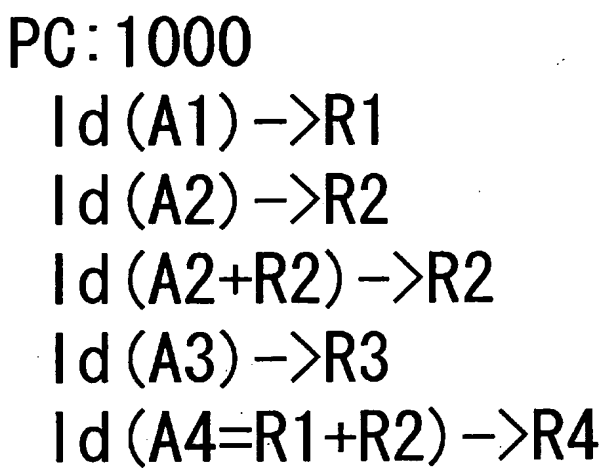

PC/Reg.
00001000
00002000
A1
A2
A3
A3
A5
A6
A6
A7
A7
A7
A7
FF
005678
01
1234
5678
88
02
5678

[図14]

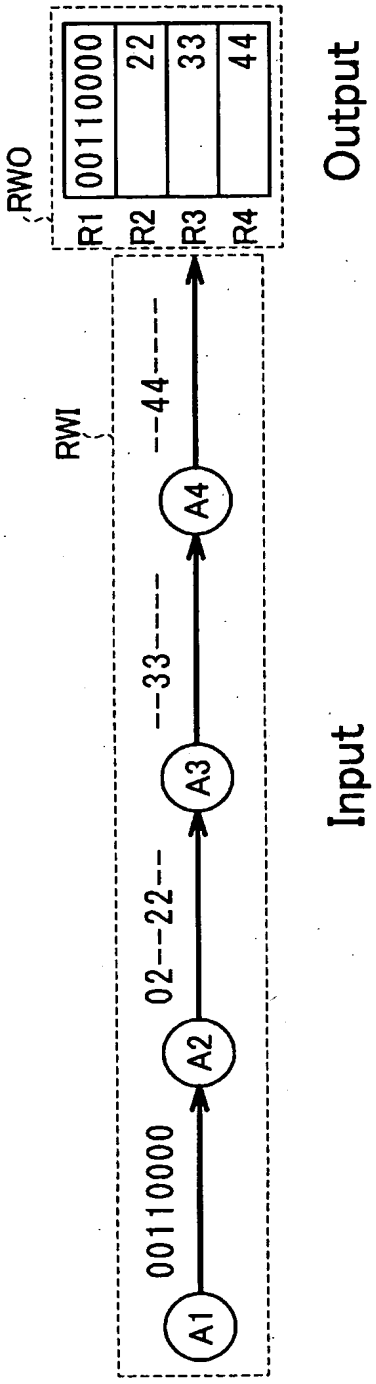

PC/Reg.
00001000
A1
----FF--
A2
--005678
--01----
00002000
A5
A6
--88----
A6
--02----
A3
1234----
PC/Reg.

A3
5678----
A7
A7
A7
5678----
A7

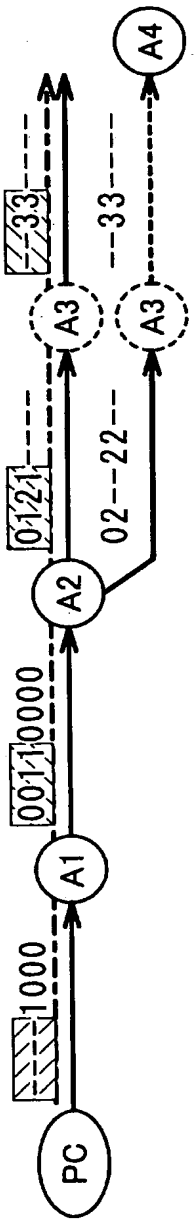
[図15]



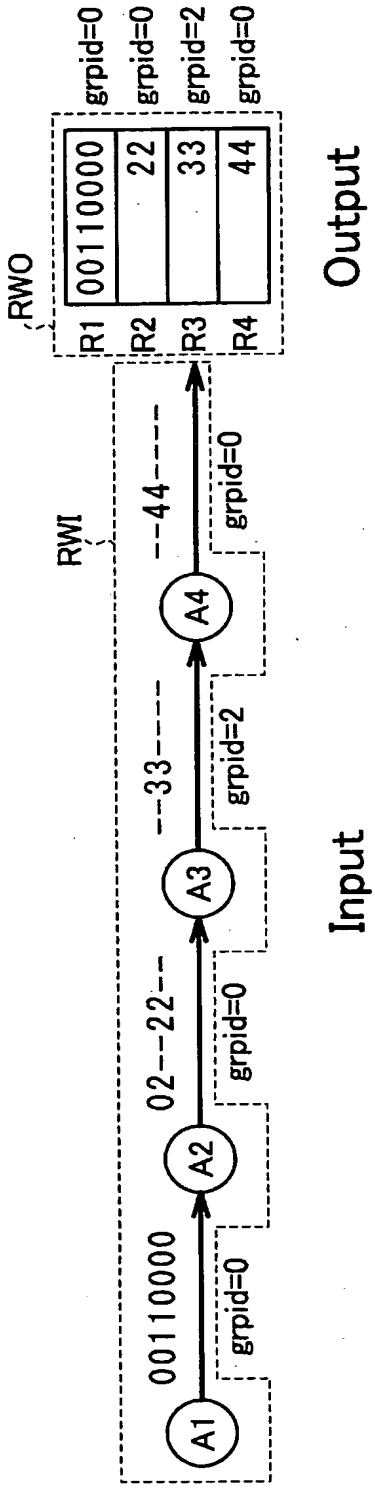
[図16]


RWO
R1
R2
R3
R4
00110000
22
33
44
Output
RWI
--44---
--33---
02--22--
00110000
A1
A2
A3
A4
Input

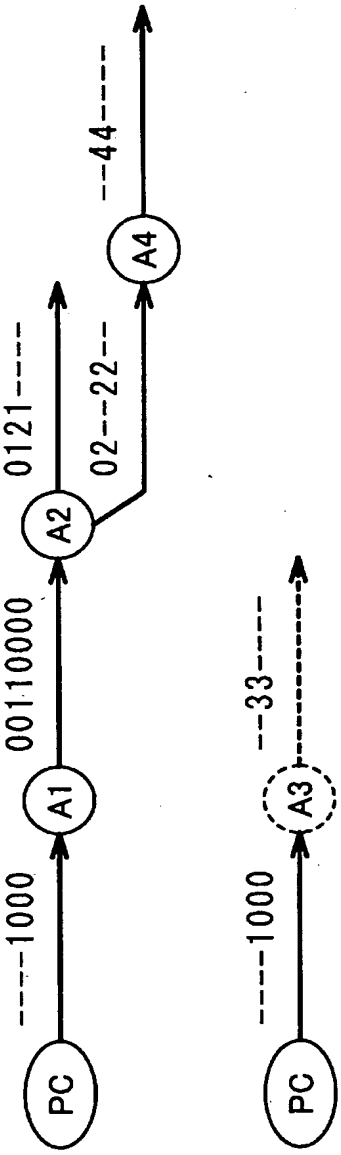
[図17]



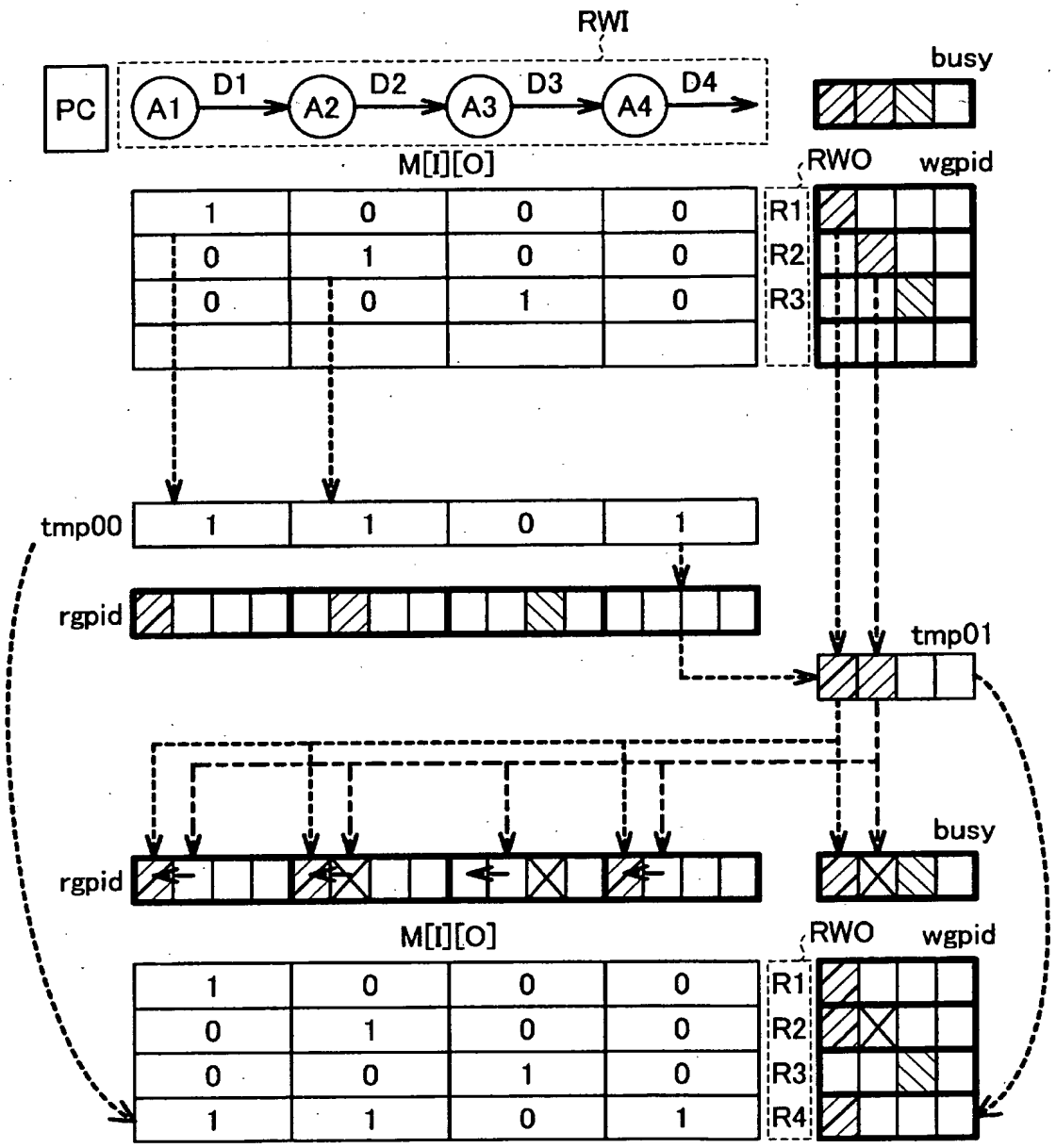
[図18]


grpid=0
grpid=0
grpid=2
grpid=0
00110000
22
33
44
RWO
R1
R2
R3
R4
Output
RWI
--44----
grpid=0
A4
--33----
grpid=2
A3
02--22--
grpid=0
A2
00110000
grpid=0
A1
Input

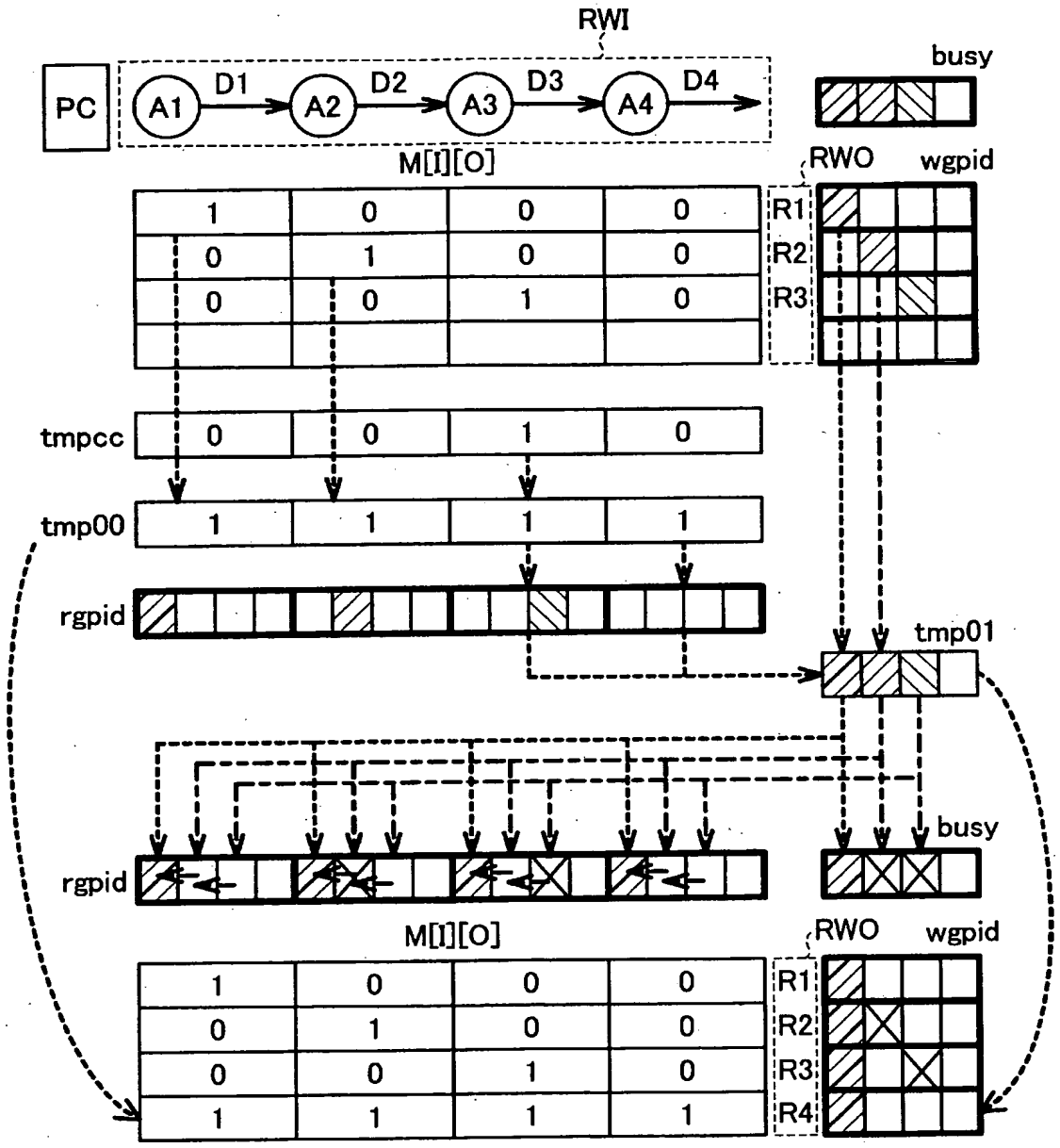
[図19]



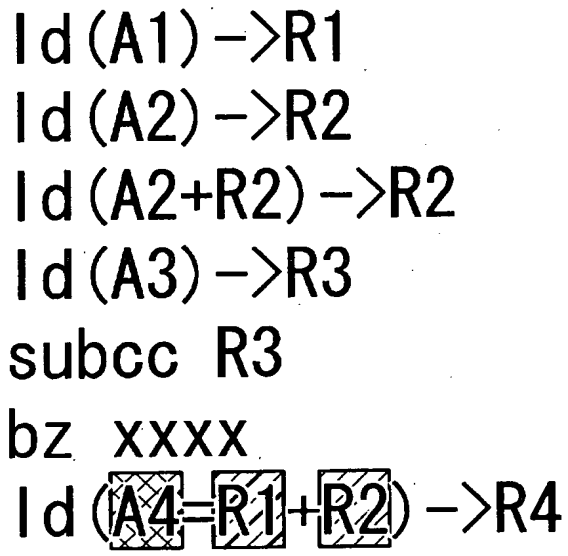
[図20]


RWI
busy
PC
A1
D1
A2
D2
A3
D3
A4
D4
M[I][O]
RWO
wgpid
1 0 0 0
0 1 0 0
0 0 1 0
R1
R2
R3
tmp00
1 1 0 1
rgpid
tmp01
busy
rgpid
M[I][O]
RWO
wgpid
1 0 0 0
0 1 0 0
0 0 1 0
1 1 0 1
R1
R2
R3
R4

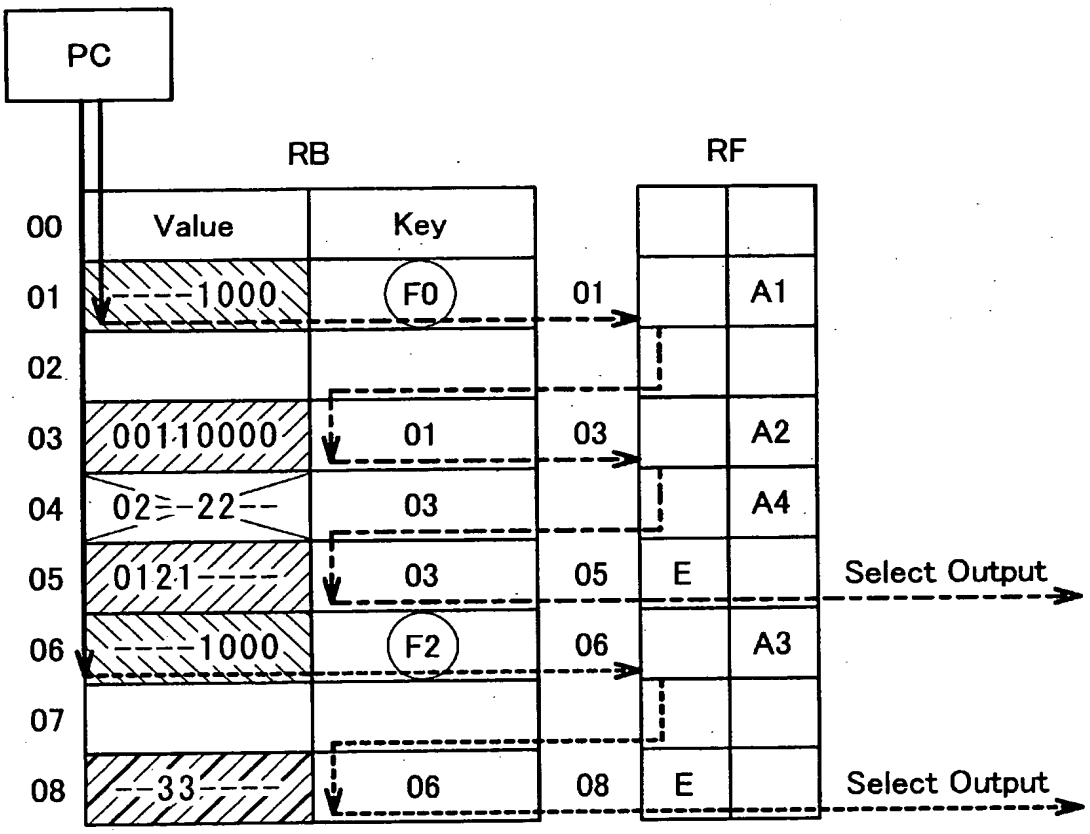
[図21]


RWI
busy
PC
A1
D1
A2
D2
A3
D3
A4
D4
M[I][O]
RWO
wgpid
1 0 0 0
0 1 0 0
0 0 1 0
R1
R2
R3
tmpcc
0 0 1 0
tmp00
1 1 1 1
rgpid
tmp01
busy
rgpid
M[I][O]
RWO
wgpid
1 0 0 0
0 1 0 0
0 0 1 0
1 1 1 1
R1
R2
R3
R4

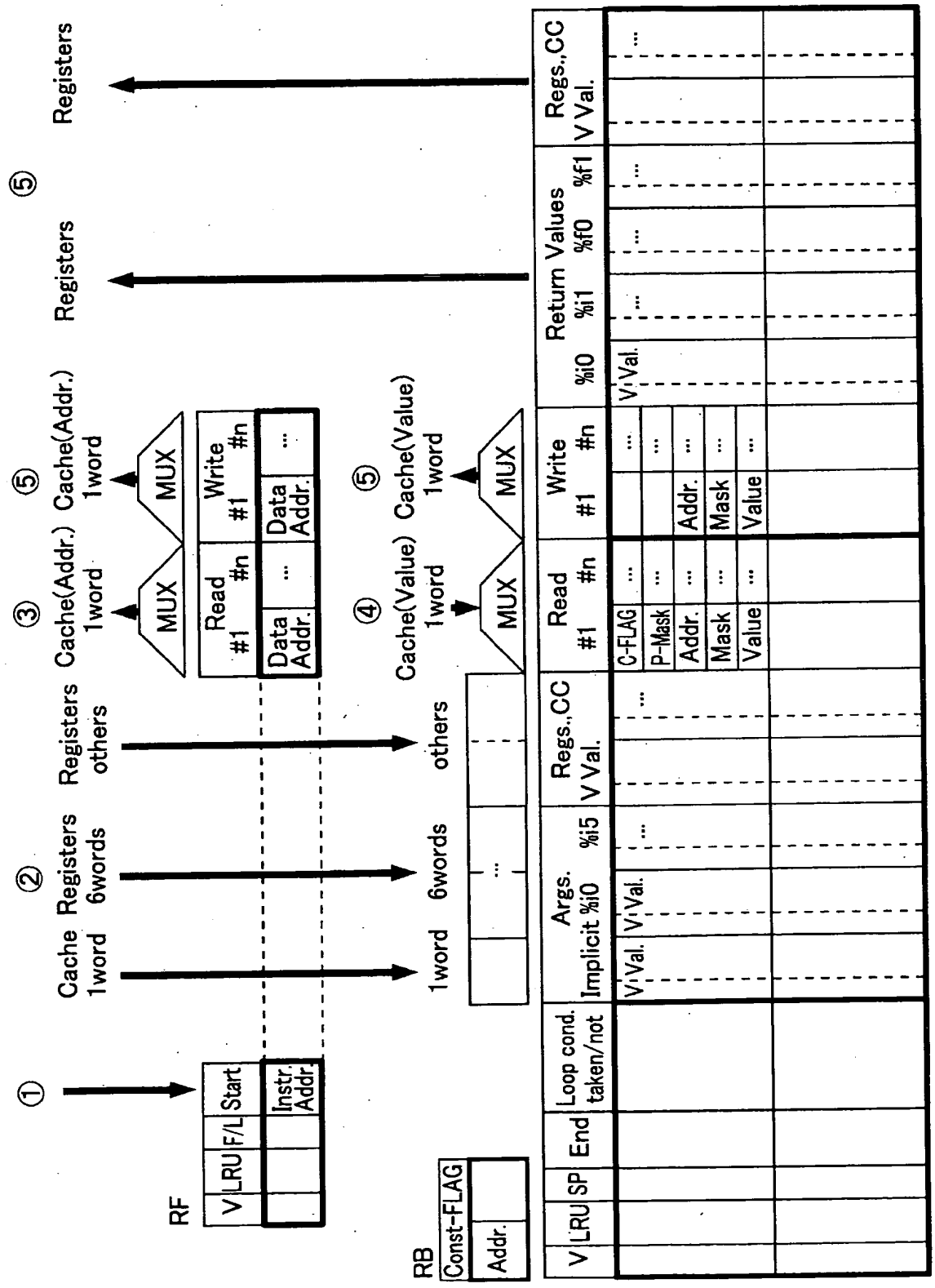
[図22]



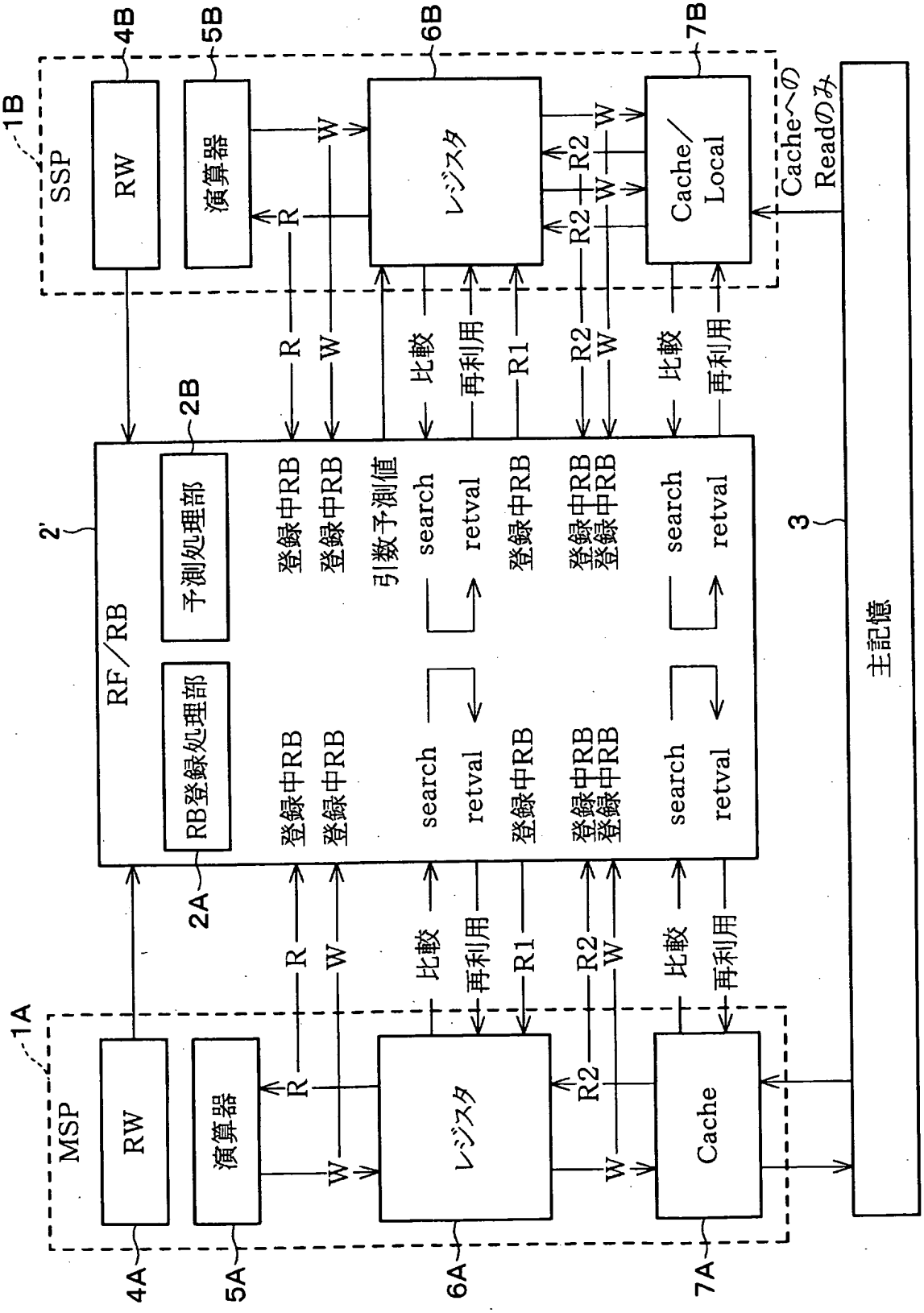
[図23]


PC
RB
RF
Value
Key
00
01
02
03
04
05
06
07
08
----1000
F0
01
A1
00110000
01
03
A2
02--22--
03
A4
0121----
03
05
E
Select Output
----1000
F2
06
A3
--33----
06
08
E
Select Output

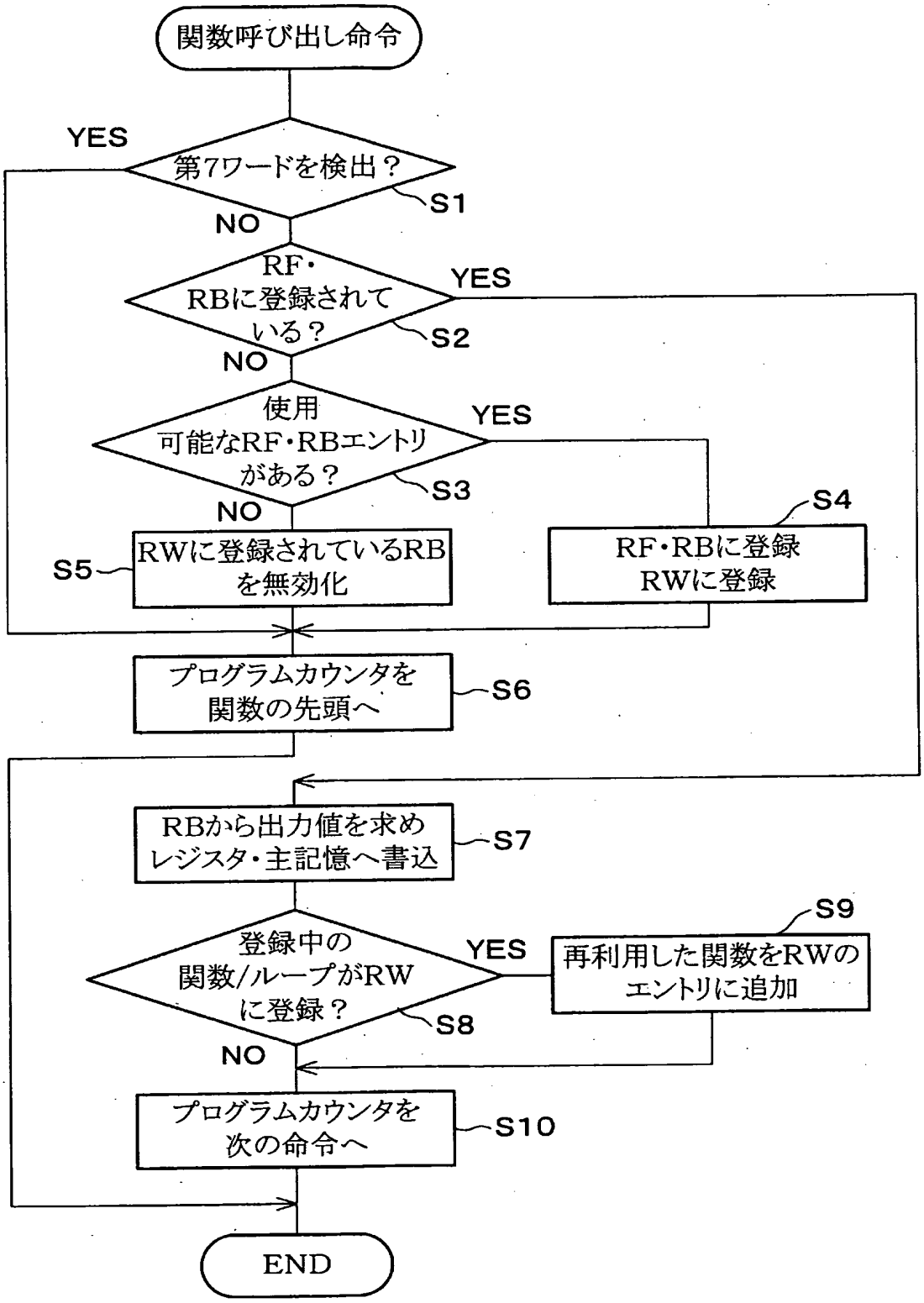
[図24]


①
②
③
④
⑤
⑤
Cache 1word
Registers 6words
Registers others
Cache(Addr.) 1word
Cache(Value) 1word
Cache(Addr.) 1word
Cache(Value) 1word
Registers
Registers
Registers
MUX
MUX
MUX
MUX
RF
V
LRU
F/L
Start
Instr. Addr
Read #1 #n
Write #1 #n
Data Addr.
RB
Const-FLAG
Addr.
V
LRU
SP
End
Loop cond. taken/not
Args. Implicit %i0
%i5
Regs.,CC V Val.
C-FLAG
P-Mask
Addr.
Mask
Value
Return Values %i0
%i1
%f0
%f1
Regs.,CC V Val.

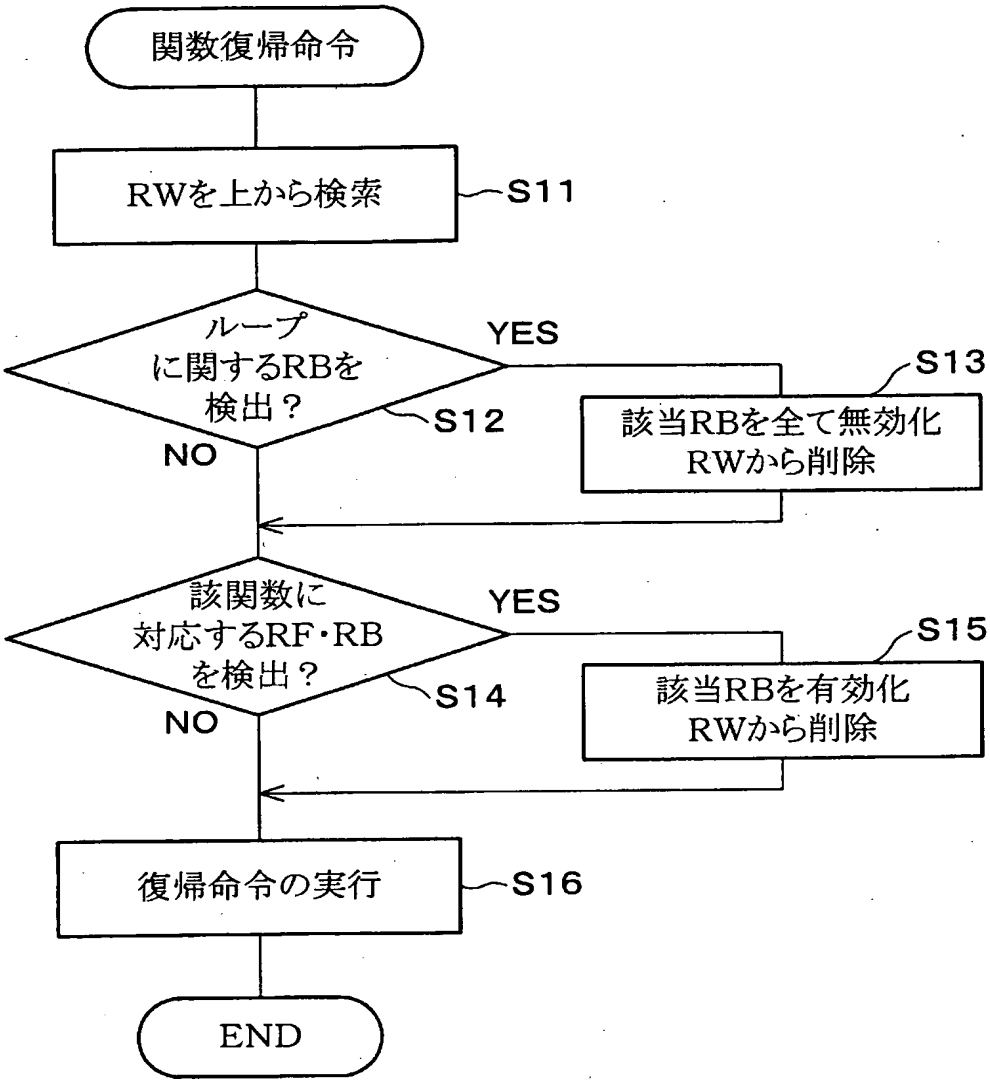
[図25]


1B
SSP
4B
RW
5B
演算器
6B
レジスタ
7B
Cache／
Local
Cacheへの
Readのみ
2'
RF／RB
2B
予測処理部
RB登録処理部
2A
登録中RB
登録中RB
引数予測値
search
retval
比較
再利用
R
W
R1
R2
3
主記憶
1A
MSP
4A
5A
6A
7A
Cache

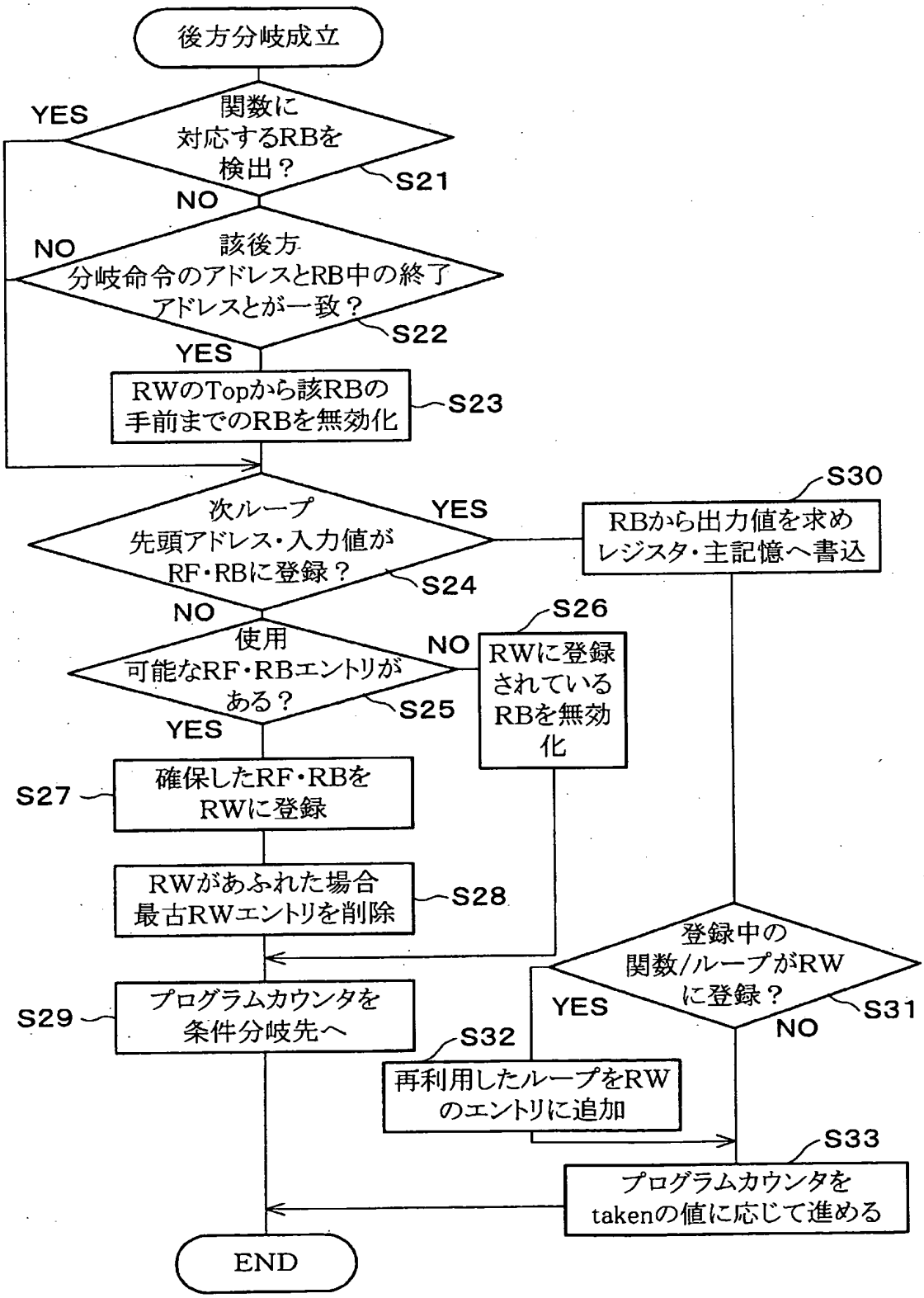
[図26]


関数呼び出し命令
YES
第7ワードを検出？
S1
NO
RF・RBに登録されている？
YES
S2
NO
使用可能なRF・RBエントリがある？
YES
S3
NO
S4
S5
RWに登録されているRBを無効化
RF・RBに登録 RWに登録
プログラムカウンタを関数の先頭へ
S6
RBから出力値を求めレジスタ・主記憶へ書込
S7
S9
登録中の関数/ループがRWに登録？
YES
再利用した関数をRWのエントリに追加
S8
NO
プログラムカウンタを次の命令へ
S10
END

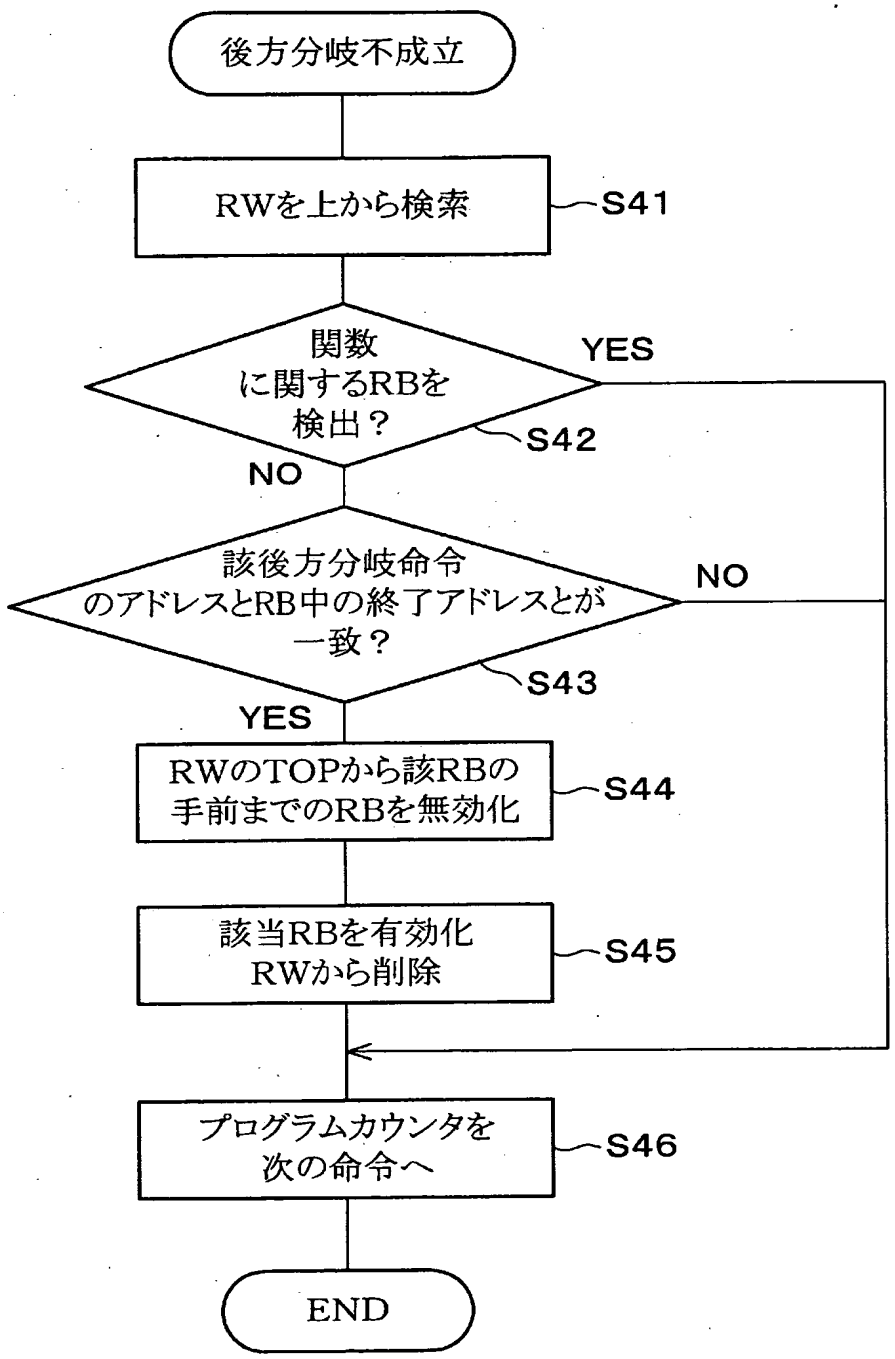
[図27]


関数復帰命令
RWを上から検索
S11
ループ
に関するRBを
検出？
S12
YES
NO
該当RBを全て無効化
RWから削除
S13
該関数に
対応するRF・RB
を検出？
S14
YES
NO
該当RBを有効化
RWから削除
S15
復帰命令の実行
S16
END

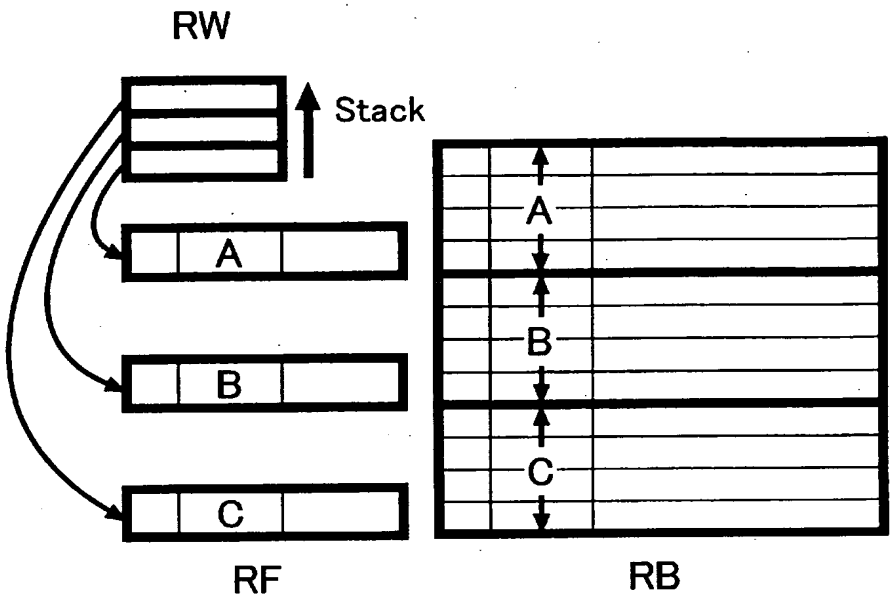
[図28]


後方分岐成立
関数に対応するRBを検出？
S21
YES
NO
該後方分岐命令のアドレスとRB中の終了アドレスとが一致？
S22
NO
YES
RWのTopから該RBの手前までのRBを無効化
S23
次ループ先頭アドレス・入力値がRF・RBに登録？
S24
YES
NO
S30
RBから出力値を求めレジスタ・主記憶へ書込
使用可能なRF・RBエントリがある？
S25
NO
YES
S26
RWに登録されているRBを無効化
S27
確保したRF・RBをRWに登録
RWがあふれた場合最古RWエントリを削除
S28
登録中の関数/ループがRWに登録？
S31
YES
NO
S29
プログラムカウンタを条件分岐先へ
S32
再利用したループをRWのエントリに追加
S33
プログラムカウンタをtakenの値に応じて進める
END

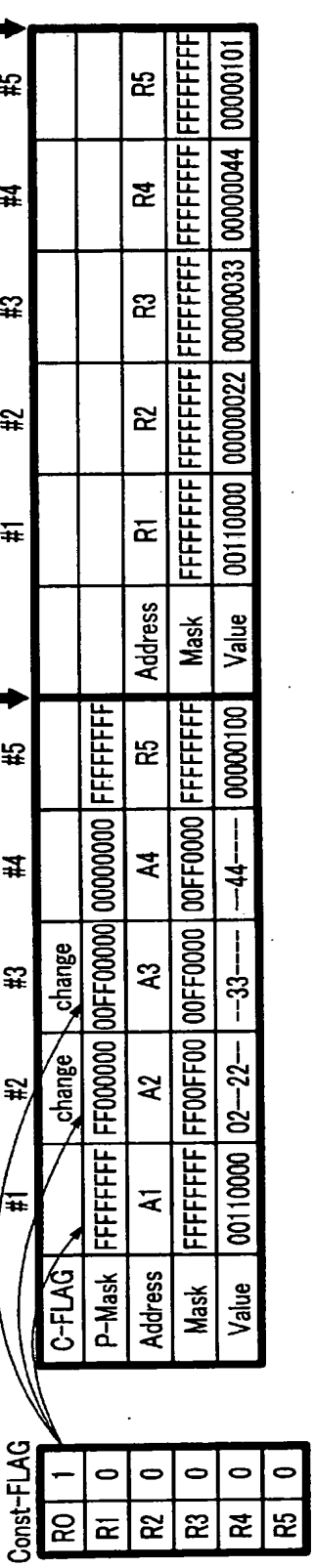
[図29]


後方分岐不成立
RWを上から検索
S41
関数
に関するRBを
検出？
YES
S42
NO
該後方分岐命令
のアドレスとRB中の終了アドレスとが
一致？
NO
S43
YES
RWのTOPから該RBの
手前までのRBを無効化
S44
該当RBを有効化
RWから削除
S45
プログラムカウンタを
次の命令へ
S46
END

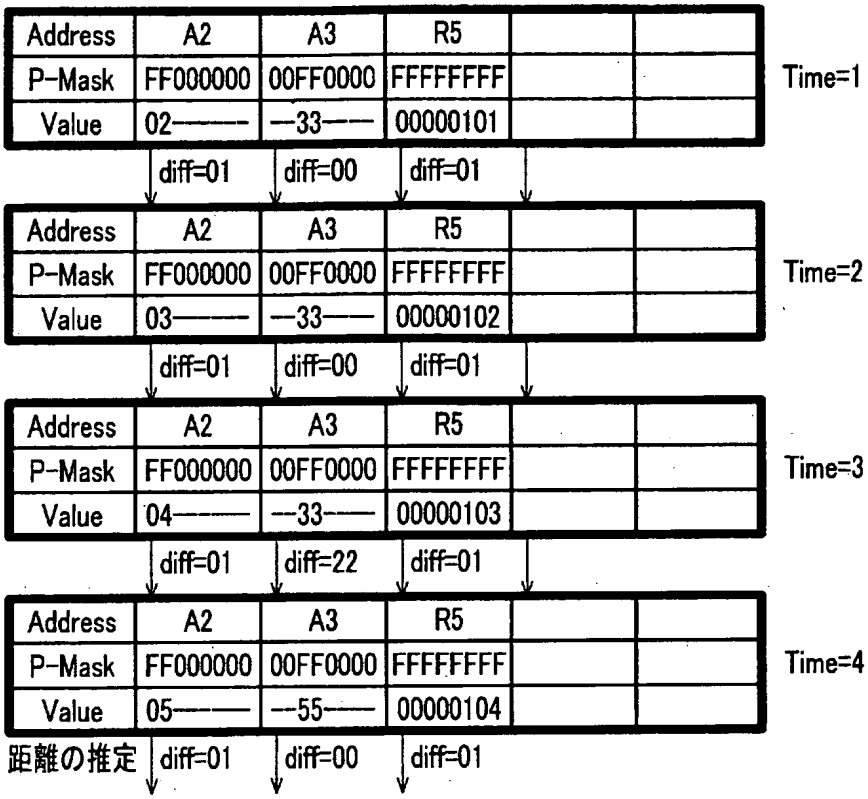
[図30]



[図31]


Const-FLAG
R0 1
R1 0
R2 0
R3 0
R4 0
R5 0
Register/Memory Read
Register/Memory Write
END
END
C-FLAG
P-Mask
Address
Mask
Value
change
change
change
#1
#2
#3
#4
#5
FFFFFFFF
FF000000
00FF0000
FF00FF00
00FF0000
00000000
00FF0000
FFFFFFFF
A1
A2
A3
A4
R5
R1
R2
R3
R4
R5
00110000
02--22--
--33----
--44----
00000100
00000022
00000033
00000044
00000101

[図32]



[図33]

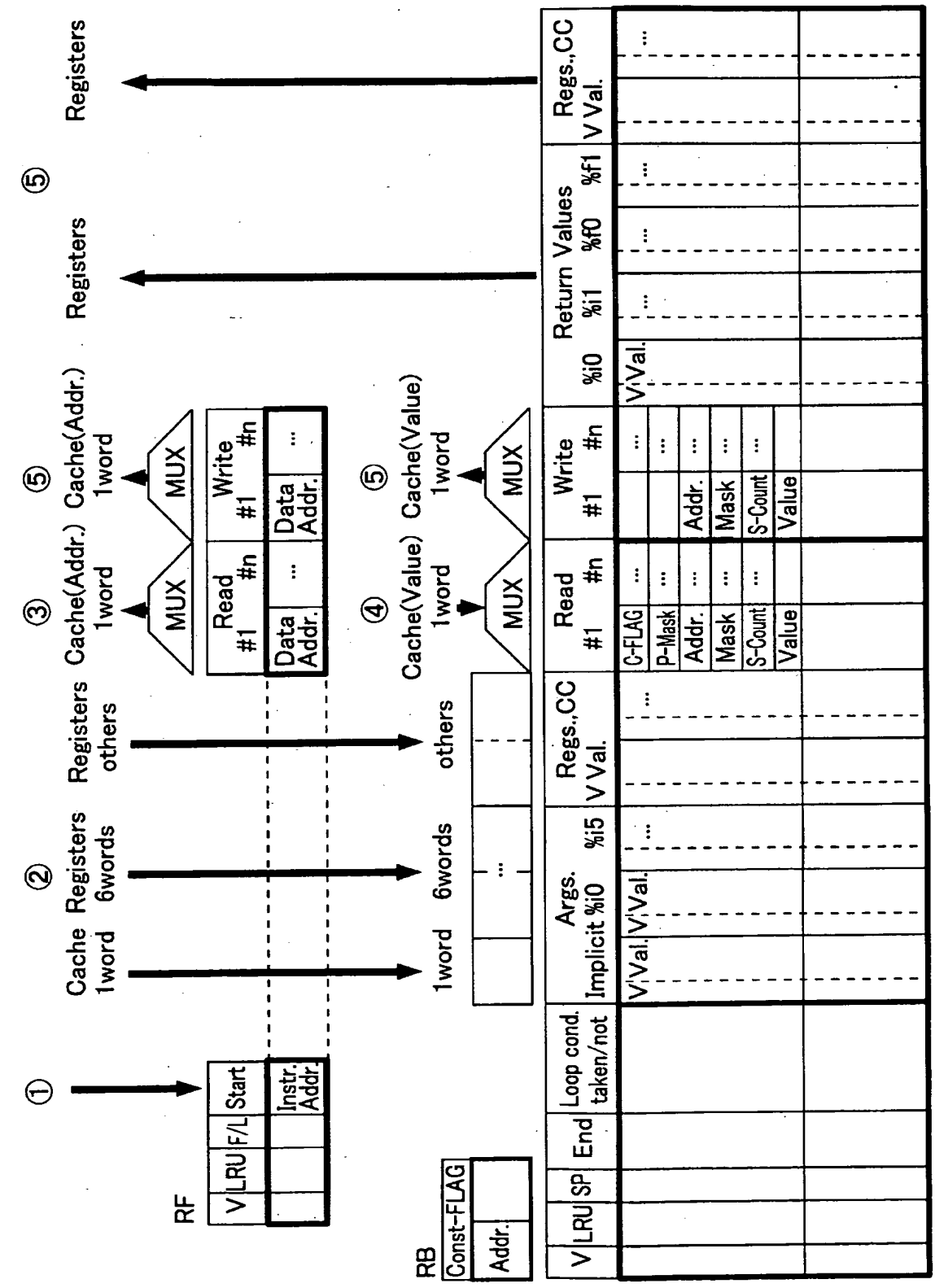
Address	A2	R5	A2+4	A3		
Mask	FF000000	FFFFFFFF	0000FF00	00FF0000		予測距離=1
Value	06———	00000105	——26—	—55——		

Address	A2	R5	A2+4	A3		
Mask	FF000000	FFFFFFFF	000000FF	00FF0000		予測距離=2
Value	07———	00000106	———27	—55——		

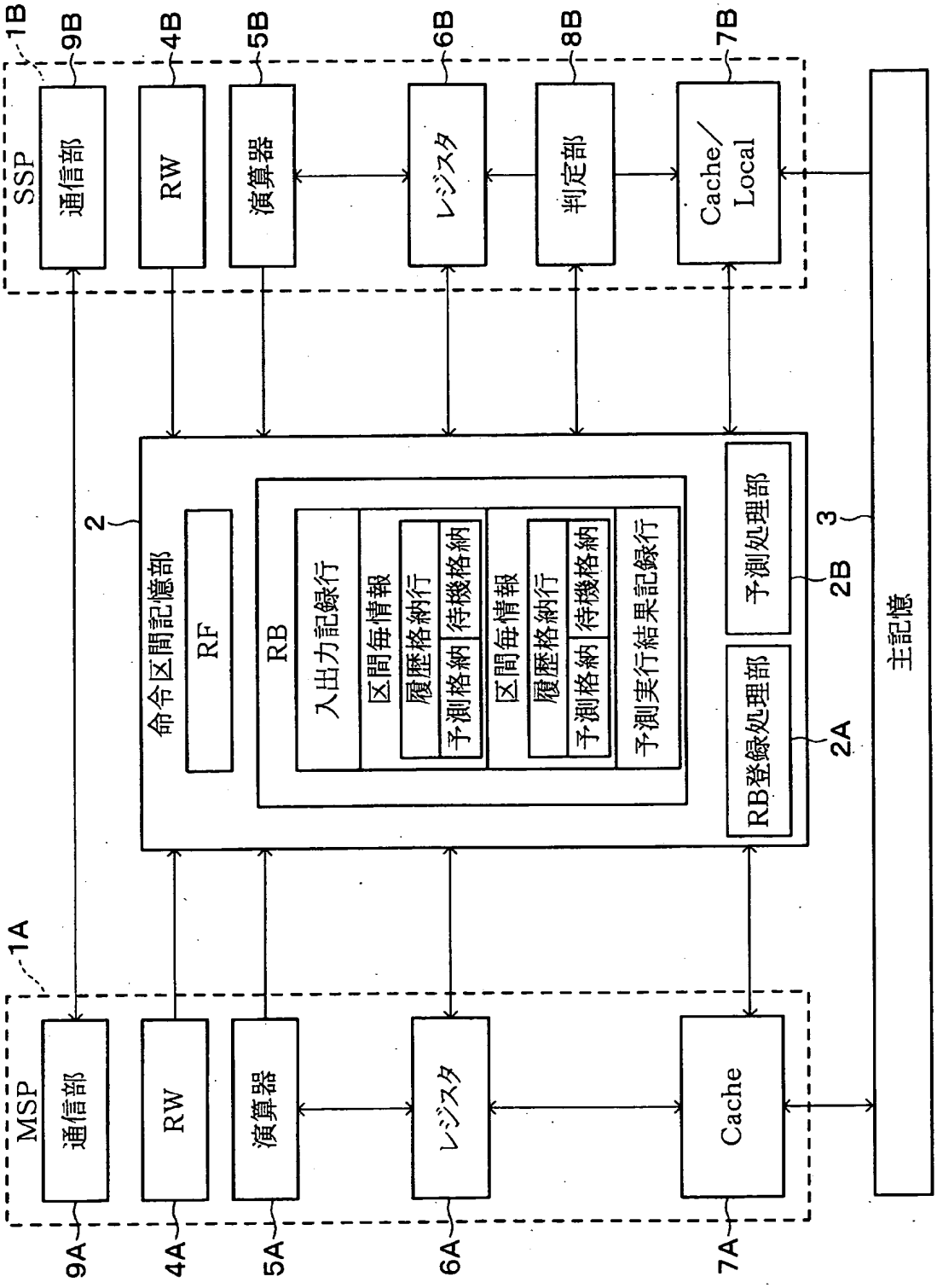
Address	A2	R5	A2+8	A3		
Mask	FF000000	FFFFFFFF	FF000000	00FF0000		予測距離=3
Value	08———	00000107	28———	—66——		

Address	A2	R5	A2+8	A3		
Mask	FF000000	FFFFFFFF	00FF0000	00FF0000		予測距離=4
Value	09———	00000108	—29——	—66——		

[図34]


①
②
③
④
⑤
⑤
⑤
Cache 1word
Registers 6words
Registers others
Cache(Addr.) 1word
Cache(Value) 1word
Cache(Addr.) 1word
Cache(Value) 1word
Registers
Registers
Registers
MUX
MUX
MUX
MUX
RF
V
LRU
F/L
Start
Instr. Addr
RB
Const-FLAG
Addr.
Read #1 #n
Data Addr.
Write #1 #n
Data Addr.
V LRU SP End Loop cond. taken/not
Args. Implicit %i0 %i5
V Val.
Regs.,CC V Val.
Read #1 #n
C-FLAG
P-Mask
Addr.
Mask
S-Count
Value
Write #1 #n
Addr.
Mask
S-Count
Value
Return Values %i0 %i1 %f0 %f1
V Val.
Regs.,CC V Val.

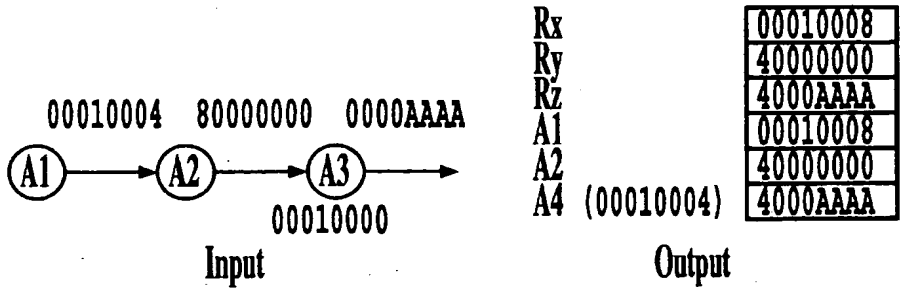
[図35]


1B
9B
4B
5B
6B
8B
7B
SSP
通信部
RW
演算器
レジスタ
判定部
Cache／Local
2
命令区間記憶部
RF
RB
入出力記録行
区間毎情報
履歴格納行
予測格納
待機格納
区間毎情報
履歴格納行
予測格納
待機格納
予測実行結果記録行
予測処理部
3
2B
RB登録処理部
2A
主記憶
1A
MSP
通信部
RW
演算器
レジスタ
Cache
9A
4A
5A
6A
7A

[図36(a)]

```
      loop:(PC=1000)
1:    set   A1          -> R1
2:    ld    (A1=R1)     -> Rx      ...(00010004)
3:    set   A2          -> R2
4:    ld    (A2=R2)     -> Ry      ...(80000000)
5:    ld    (A3=Rx-4)   -> Rz      ...(0000aaaa)
6:    add   Rx+4        -> Rx      ... 00010008
7:    st    Rx          ->(A1=R1)  ... 00010008
8:    shift Ry          -> Ry      ... 40000000
9:    st    Ry          ->(A2=R2)  ... 40000000
10:   add   Ry+Rz       -> Rz      ... 4000aaaa
11:   st    Rz          ->(A4=Rx)  ... 4000aaaa
12:   br    loop
```

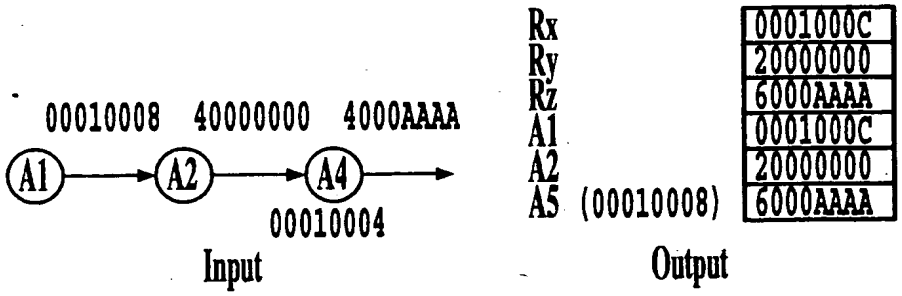
[図36(b)]



[図36(c)]

```
loop:(PC=1000)
set   A1        -> R1
ld    (A1=R1)   -> Rx     ...(00010008)
set   A2        -> R2
ld    (A2=R2)   -> Ry     ...(40000000)
ld    (A4=Rx-4) -> Rz     ...(4000aaaa)
add   Rx+4      -> Rx     ... 0001000c
st    Rx        ->(A1=R1) ... 0001000c
shift Ry        -> Ry     ... 20000000
st    Ry        ->(A2=R2) ... 20000000
add   Ry+Rz     -> Rz     ... 6000aaaa
st    Rz        ->(A5=Rx) ... 6000aaaa
br    loop
```

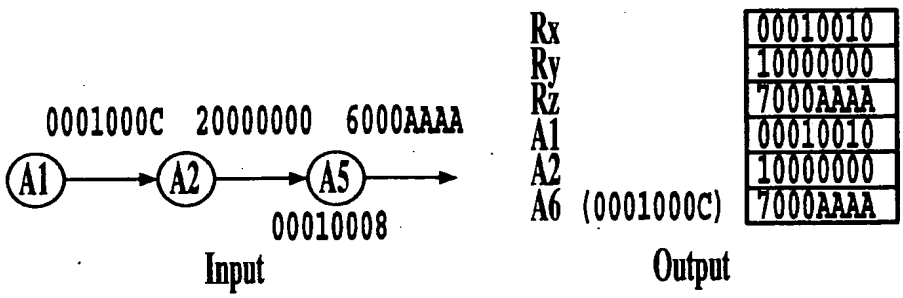
[図36(d)]



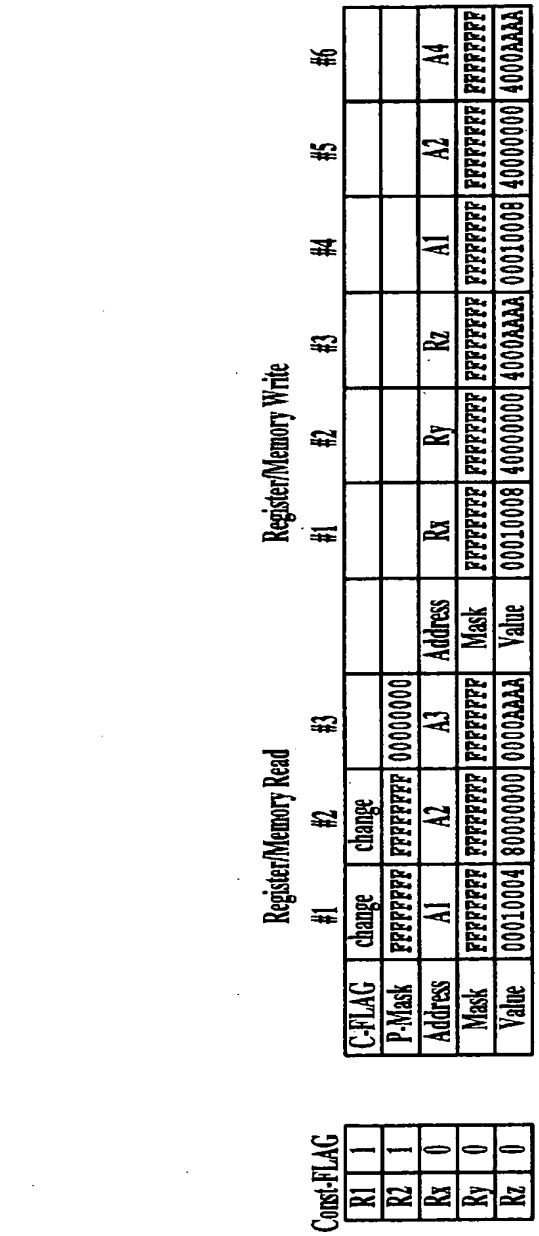
[図36(e)]

```
loop:(PC=1000)
set   A1        -> R1
ld    (A1=R1)   -> Rx     ...(0001000c)
set   A2        -> R2
ld    (A2=R2)   -> Ry     ...(20000000)
ld    (A5=Rx-4) -> Rz     ...(6000aaaa)
add   Rx+4      -> Rx     ... 00010010
st    Rx        ->(A1=R1) ... 00010010
shift Ry        -> Ry     ... 10000000
st    Ry        ->(A2=R2) ... 10000000
add   Ry+Rz     -> Rz     ... 7000aaaa
st    Rz        ->(A6=Rx) ... 7000aaaa
br    loop
```

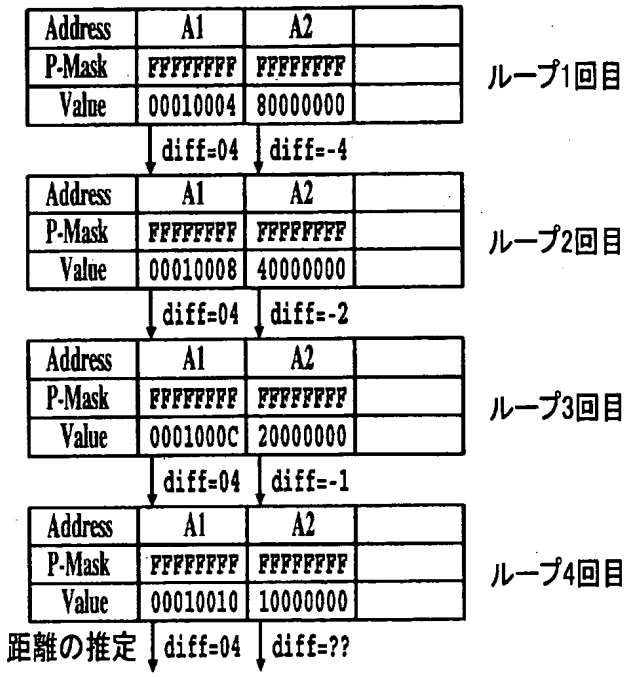
[図36(f)]



[図37]


Register/Memory Read
Register/Memory Write
Const-FLAG
R1 1
R2 1
Rx 0
Ry 0
Rz 0
C-FLAG change change
P-Mask FFFFFFFF FFFFFFFF 00000000
Address A1 A2 A3 Address Rx Ry Rz A1 A2 A4
Mask FFFFFFFF FFFFFFFF FFFFFFFF Mask FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
Value 00010004 80000000 0000AAAA Value 00010008 40000000 4000AAAA 00010008 40000000 4000AAAA
#1 #2 #3 #1 #2 #3 #4 #5 #6

[図38(a)]



[図38(b)]

予測値格納領域　　待機要アドレス格納領域

Address	A1	A2	A7
Mask	FFFFFFFF	FFFFFFFF	FFFFFFFF
Value	00010014	10000000	????????

ループ5回目
予測距離=1⇒MSPへ割当

Address	A1	A2	A8
Mask	FFFFFFFF	FFFFFFFF	FFFFFFFF
Value	00010018	10000000	????????

ループ6回目
予測距離=2⇒SSP#1へ割当

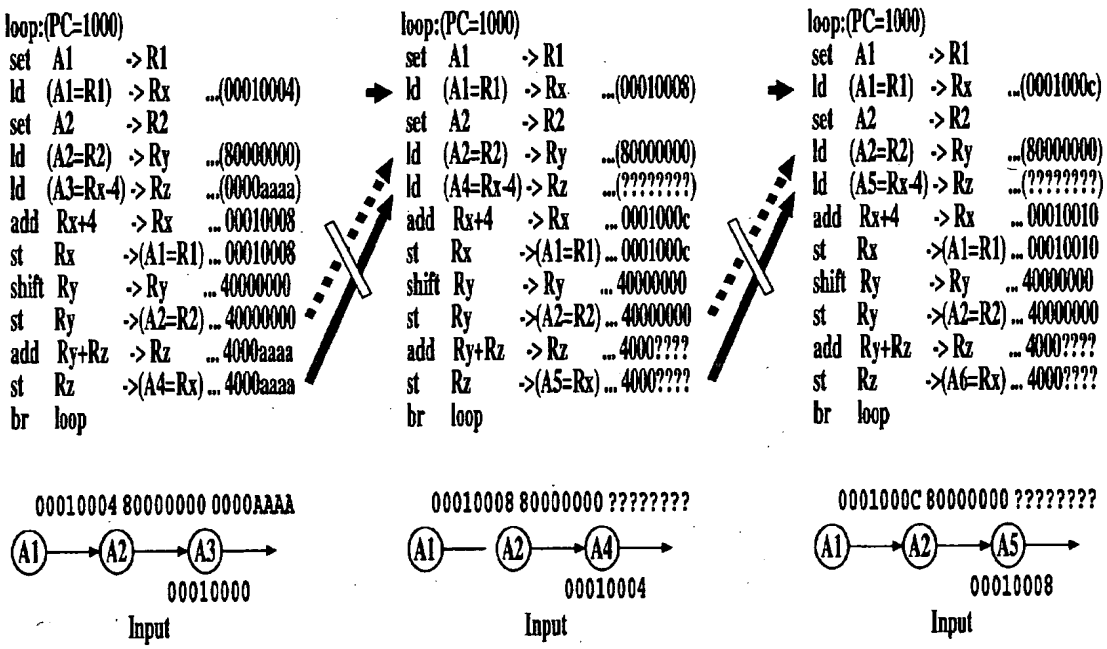
Address	A1	A2	A9
Mask	FFFFFFFF	FFFFFFFF	FFFFFFFF
Value	0001001C	10000000	????????

ループ7回目
予測距離=3⇒SSP#2へ割当

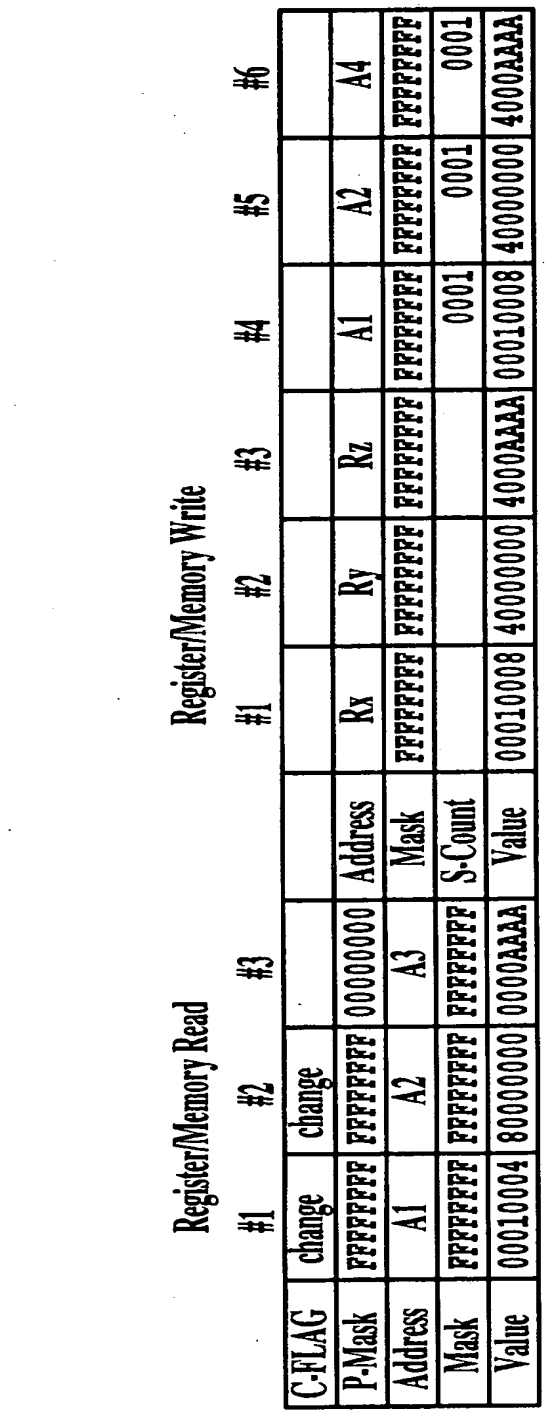
Address	A1	A2	A10
Mask	FFFFFFFF	FFFFFFFF	FFFFFFFF
Value	00010020	10000000	????????

ループ8回目
予測距離=4⇒SSP#3へ割当

[図39]


loop:(PC=1000)
set A1 -> R1
ld (A1=R1) -> Rx ...(00010004)
set A2 -> R2
ld (A2=R2) -> Ry ...(80000000)
ld (A3=Rx-4) -> Rz ...(0000aaaa)
add Rx+4 -> Rx ... 00010008
st Rx ->(A1=R1) ... 00010008
shift Ry -> Ry ... 40000000
st Ry ->(A2=R2) ... 40000000
add Ry+Rz -> Rz ... 4000aaaa
st Rz ->(A4=Rx) ... 4000aaaa
br loop
loop:(PC=1000)
set A1 -> R1
ld (A1=R1) -> Rx ...(00010008)
set A2 -> R2
ld (A2=R2) -> Ry ...(80000000)
ld (A4=Rx-4) -> Rz ...(????????)
add Rx+4 -> Rx ... 0001000c
st Rx ->(A1=R1) ... 0001000c
shift Ry -> Ry ... 40000000
st Ry ->(A2=R2) ... 40000000
add Ry+Rz -> Rz ... 4000????
st Rz ->(A5=Rx) ... 4000????
br loop
loop:(PC=1000)
set A1 -> R1
ld (A1=R1) -> Rx ...(0001000c)
set A2 -> R2
ld (A2=R2) -> Ry ...(80000000)
ld (A5=Rx-4) -> Rz ...(????????)
add Rx+4 -> Rx ... 00010010
st Rx ->(A1=R1) ... 00010010
shift Ry -> Ry ... 40000000
st Ry ->(A2=R2) ... 40000000
add Ry+Rz -> Rz ... 4000????
st Rz ->(A6=Rx) ... 4000????
br loop
00010004 80000000 0000AAAA
A1
A2
A3
00010000
Input
00010008 80000000 ????????
A1
A2
A4
00010004
Input
0001000C 80000000 ????????
A1
A2
A5
00010008
Input

[図40(a)]



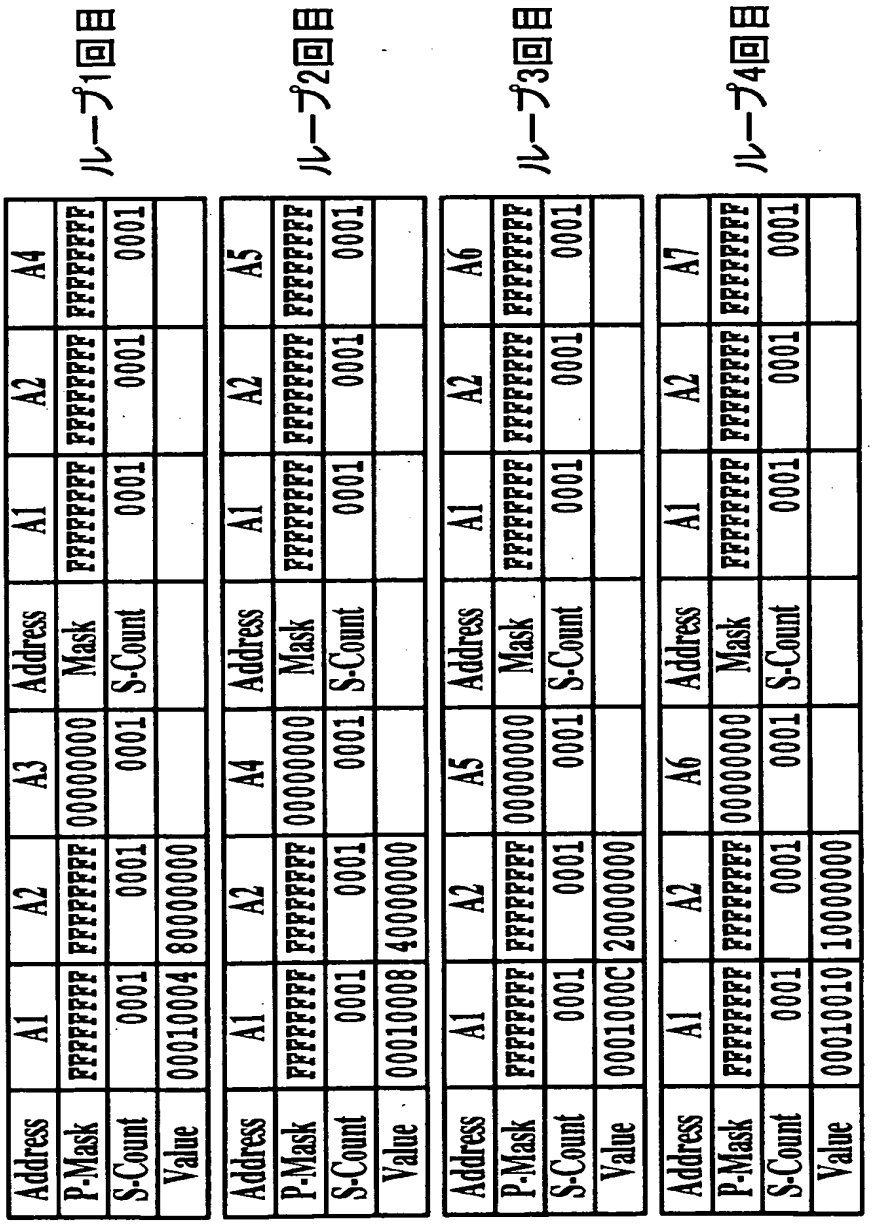
Register/Memory Read

	#1	#2	#3
C-FLAG	change	change	
P-Mask	FFFFFFFF	FFFFFFFF	00000000
Address	A1	A2	A3
Mask	FFFFFFFF	FFFFFFFF	FFFFFFFF
Value	00010004	80000000	0000AAAA

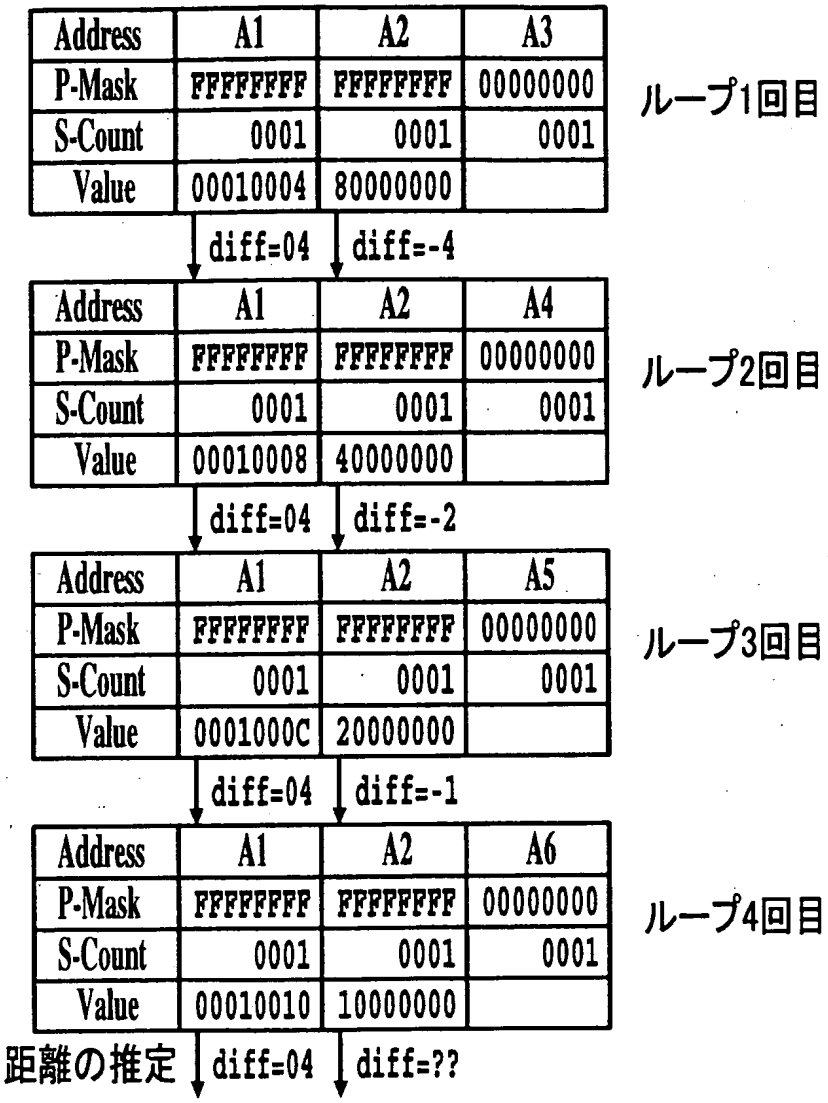
Register/Memory Write

	#1	#2	#3	#4	#5	#6
	Rx	Ry	Rz	A1	A2	A4
Address	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
Mask				0001	0001	0001
S-Count						
Value	00010008	40000000	4000AAAA	00010008	40000000	4000AAAA

[図40(b)]


ループ1回目
Address A1 A2 A3 Address A1 A2 A4
P-Mask FFFFFFFF FFFFFFFF 00000000 Mask FFFFFFFF FFFFFFFF FFFFFFFF
S-Count 0001 0001 0001 S-Count 0001 0001 0001
Value 00010004 80000000
ループ2回目
Address A1 A2 A4 Address A1 A2 A5
P-Mask FFFFFFFF FFFFFFFF 00000000 Mask FFFFFFFF FFFFFFFF FFFFFFFF
S-Count 0001 0001 0001 S-Count 0001 0001 0001
Value 00010008 40000000
ループ3回目
Address A1 A2 A5 Address A1 A2 A6
P-Mask FFFFFFFF FFFFFFFF 00000000 Mask FFFFFFFF FFFFFFFF FFFFFFFF
S-Count 0001 0001 0001 S-Count 0001 0001 0001
Value 0001000C 20000000
ループ4回目
Address A1 A2 A6 Address A1 A2 A7
P-Mask FFFFFFFF FFFFFFFF 00000000 Mask FFFFFFFF FFFFFFFF FFFFFFFF
S-Count 0001 0001 0001 S-Count 0001 0001 0001
Value 00010010 10000000

[図41(a)]


Address A1 A2 A3
P-Mask FFFFFFFF FFFFFFFF 00000000
S-Count 0001 0001 0001
Value 00010004 80000000
ループ1回目
diff=04 diff=-4
Address A1 A2 A4
P-Mask FFFFFFFF FFFFFFFF 00000000
S-Count 0001 0001 0001
Value 00010008 40000000
ループ2回目
diff=04 diff=-2
Address A1 A2 A5
P-Mask FFFFFFFF FFFFFFFF 00000000
S-Count 0001 0001 0001
Value 0001000C 20000000
ループ3回目
diff=04 diff=-1
Address A1 A2 A6
P-Mask FFFFFFFF FFFFFFFF 00000000
S-Count 0001 0001 0001
Value 00010010 10000000
ループ4回目
距離の推定
diff=04 diff=??

[図41(b)]

予測値格納領域　　待機要アドレス格納領域

Address	A1	A2	A7
Mask	FFFFFFFF	FFFFFFFF	FFFFFFFF
S-Count		0000	0001
Value	00010014	WAIT	WAIT

ループ5回目
予測距離=1⇒MSPへ割当

Address	A1	A2	A8
Mask	FFFFFFFF	FFFFFFFF	FFFFFFFF
S-Count		0001	0001
Value	00010018	WAIT	WAIT

ループ6回目
予測距離=2⇒SSP#1へ割当

Address	A1	A2	A9
Mask	FFFFFFFF	FFFFFFFF	FFFFFFFF
S-Count		0002	0001
Value	0001001C	WAIT	WAIT

ループ7回目
予測距離=3⇒SSP#2へ割当

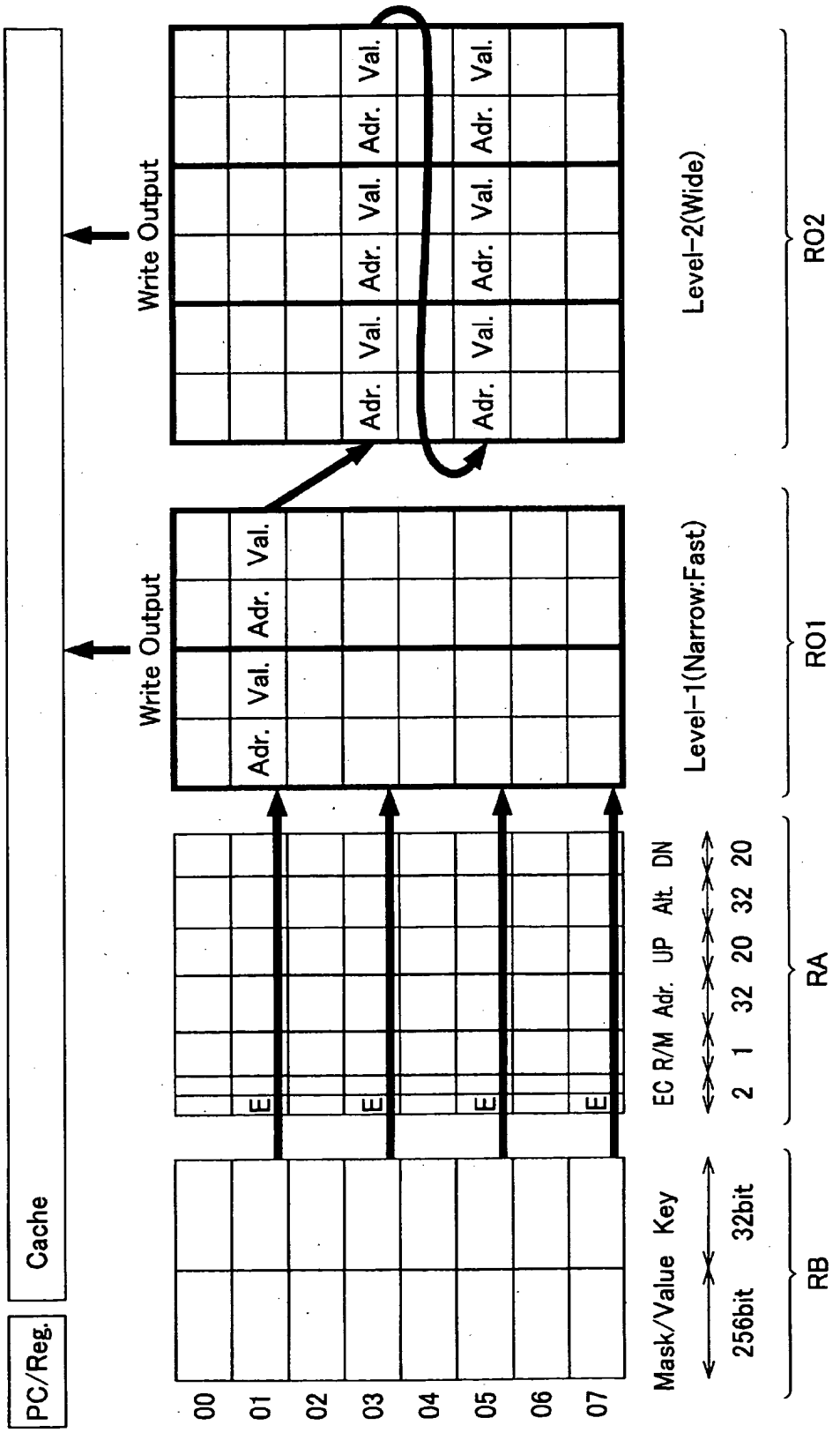
Address	A1	A2	A10
Mask	FFFFFFFF	FFFFFFFF	FFFFFFFF
S-Count		0003	0001
Value	00010020	WAIT	WAIT

ループ8回目
予測距離=4⇒SSP#3へ割当

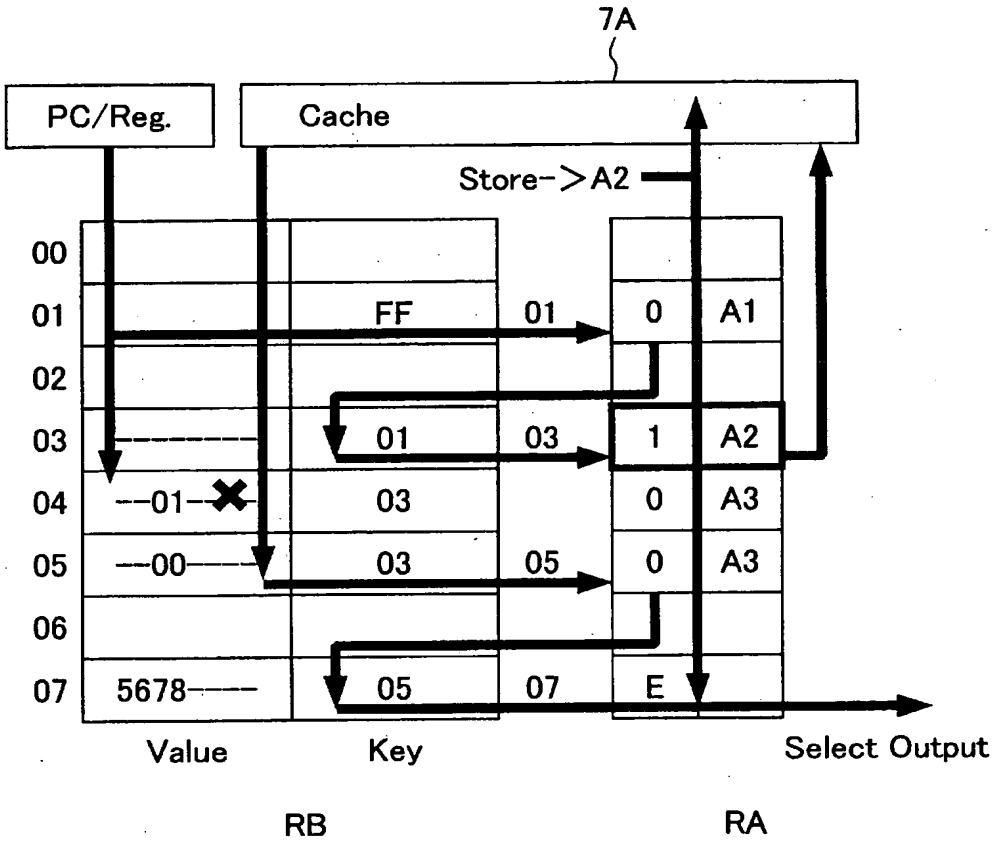
[図42]

ループ5回目 (MSP)
loop:(PC=1000)
set A1 -> R1
ld (A1=R1) -> Rx ...(00010014)
set A2 -> R2
ld (A2=R2) -> Ry ...(08000000)
ld (A7=Rx-4) -> Rz ...(780aaaaa)
add Rx+4 -> Rx ... 00010018
st Rx ->(A1=R1) ... 00010018
shift Ry -> Ry ... 04000000
st Ry ->(A2=R2) ... 04000000
add Ry+Rz -> Rz ... 7c00aaaa
st Rz ->(A8=Rx) ... 7c00aaaa
br loop
ループ6回目 (SSP#1)
Address A1 A2 A8
Mask FFFFFFFF FFFFFFFF FFFFFFFF
S-Count 0001 0001
Value 00010018 WAIT WAIT
loop:(PC=1000)
set A1 -> R1
ld (A1=R1) -> Rx ...(00010018)
set A2 -> R2
Address A1 A2 A8
Mask FFFFFFFF FFFFFFFF FFFFFFFF
S-Count 0000 0000
Value 00010018 04000000 7C00AAAA
ld (A2=R2) -> Ry ...(04000000)
ld (A8=Rx-4) -> Rz ...(7c00aaaa)
add Rx+4 -> Rx ... 0001001c
st Rx ->(A1=R1) ... 0001001c
shift Ry -> Ry ... 02000000
st Ry ->(A2=R2) ... 02000000
add Ry+Rz -> Rz ... 7e00aaaa
st Rz ->(A9=Rx) ... 7e00aaaa
br loop
00010018 04000000 7C00AAAA
A1 A2 A4
00010014
Input
Rx 0001001C
Ry 02000000
Rz 7E00AAAA
A1 0001001C
A2 02000000
A9 7E00AAAA
Output
ループ7回目 (SSP#2)
Address A1 A2 A9
Mask FFFFFFFF FFFFFFFF FFFFFFFF
S-Count 0002 0001
Value 0001001C WAIT WAIT
loop:(PC=1000)
set A1 -> R1
ld (A1=R1) -> Rx ...(0001001c)
set A2 -> R2
Address A1 A2 A9
Mask FFFFFFFF FFFFFFFF FFFFFFFF
S-Count 0000 0000
Value 0001001C 02000000 7E00AAAA
ld (A2=R2) -> Ry ...(02000000)
ld (A9=Rx-4) -> Rz ...(7e00aaaa)
add Rx+4 -> Rx ... 00010020
st Rx ->(A1=R1) ... 00010020
shift Ry -> Ry ... 01000000
st Ry ->(A2=R2) ... 01000000
add Ry+Rz -> Rz ... 7f00aaaa
st Rz ->(A10=Rx) ... 7f00aaaa
br loop
0001001C 02000000 7E00AAAA
A1 A2 A5
00010018
Input
Rx 00010020
Ry 01000000
Rz 7F00AAAA
A1 00010020
A2 01000000
A10 7F00AAAA
Output

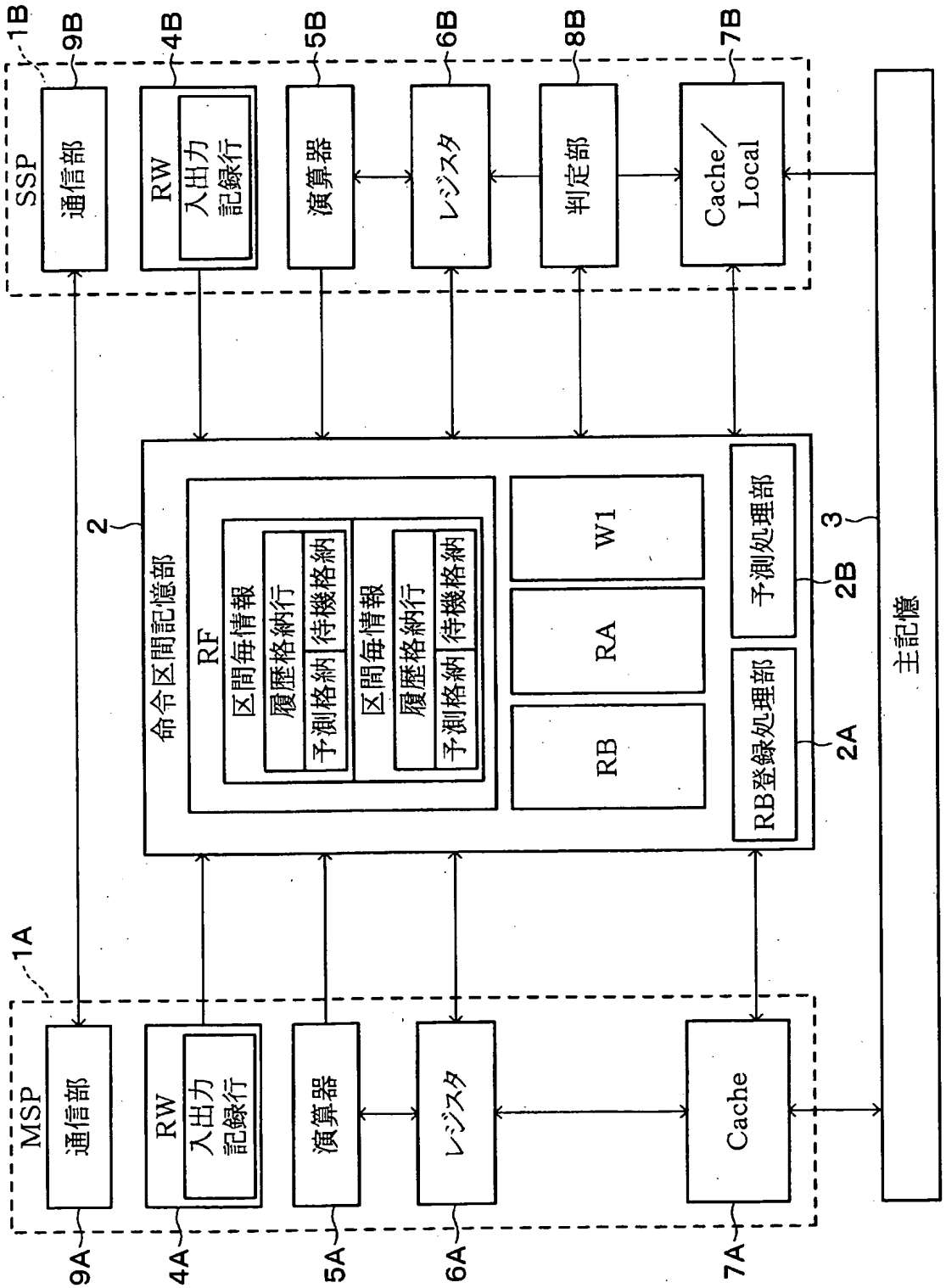
[図43]


PC/Reg.
Cache
Write Output
Write Output
Adr.
Val.
E
00
01
02
03
04
05
06
07
Mask/Value Key
256bit 32bit
EC R/M Adr. UP Alt. DN
2 1 32 20 32 20
Level-1(Narrow:Fast)
Level-2(Wide)
RB
RA
RO1
RO2

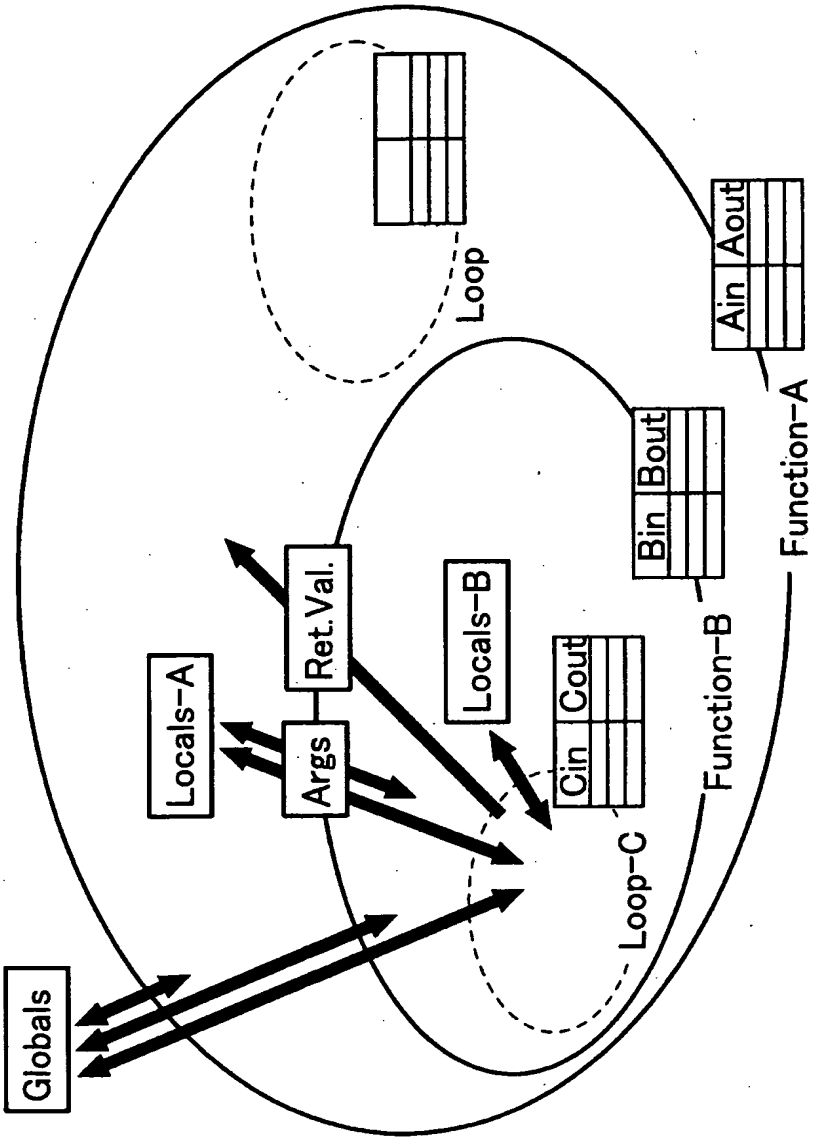
[図44]


7A
PC/Reg.
Cache
Store->A2
00
01
02
03
04
05
06
07
FF
01
03
03
03
05
05
07
0
A1
1
A2
0
A3
0
A3
E
--01--
--00--
5678----
Value
Key
Select Output
RB
RA

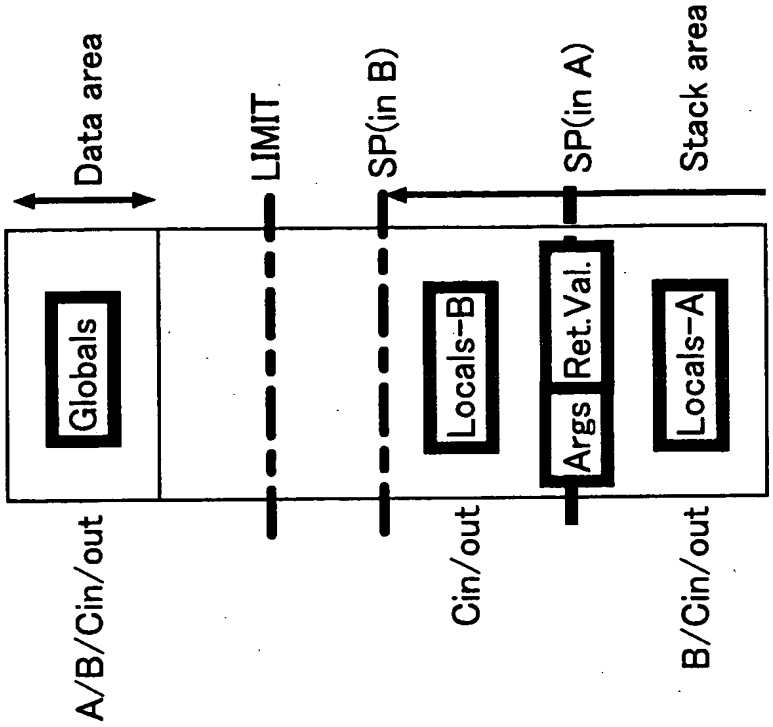
[図45]


1B
9B
4B
5B
6B
8B
7B
SSP
通信部
RW
入出力
記録行
演算器
レジスタ
判定部
Cache/
Local
2
命令区間記憶部
RF
区間毎情報
履歴格納行
予測格納
待機格納
区間毎情報
履歴格納行
予測格納
待機格納
W1
RA
RB
予測処理部
RB登録処理部
3
2B
2A
主記憶
1A
MSP
通信部
RW
入出力
記録行
演算器
レジスタ
Cache
9A
4A
5A
6A
7A

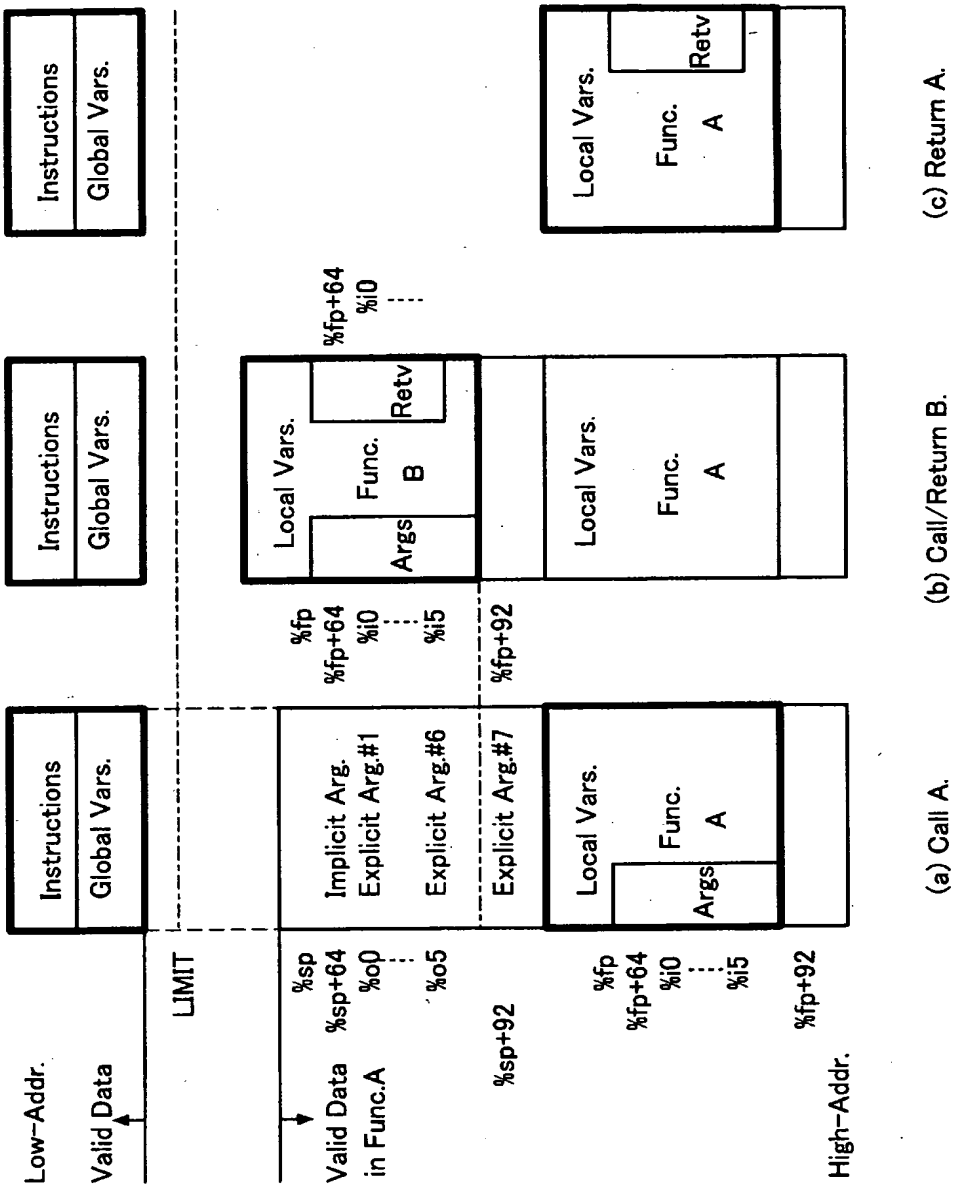
[図46(a)]


Globals
Locals-A
Ret.Val.
Args
Locals-B
Cout
Cin
Loop-C
Loop
Bin
Bout
Function-B
Ain
Aout
Function-A

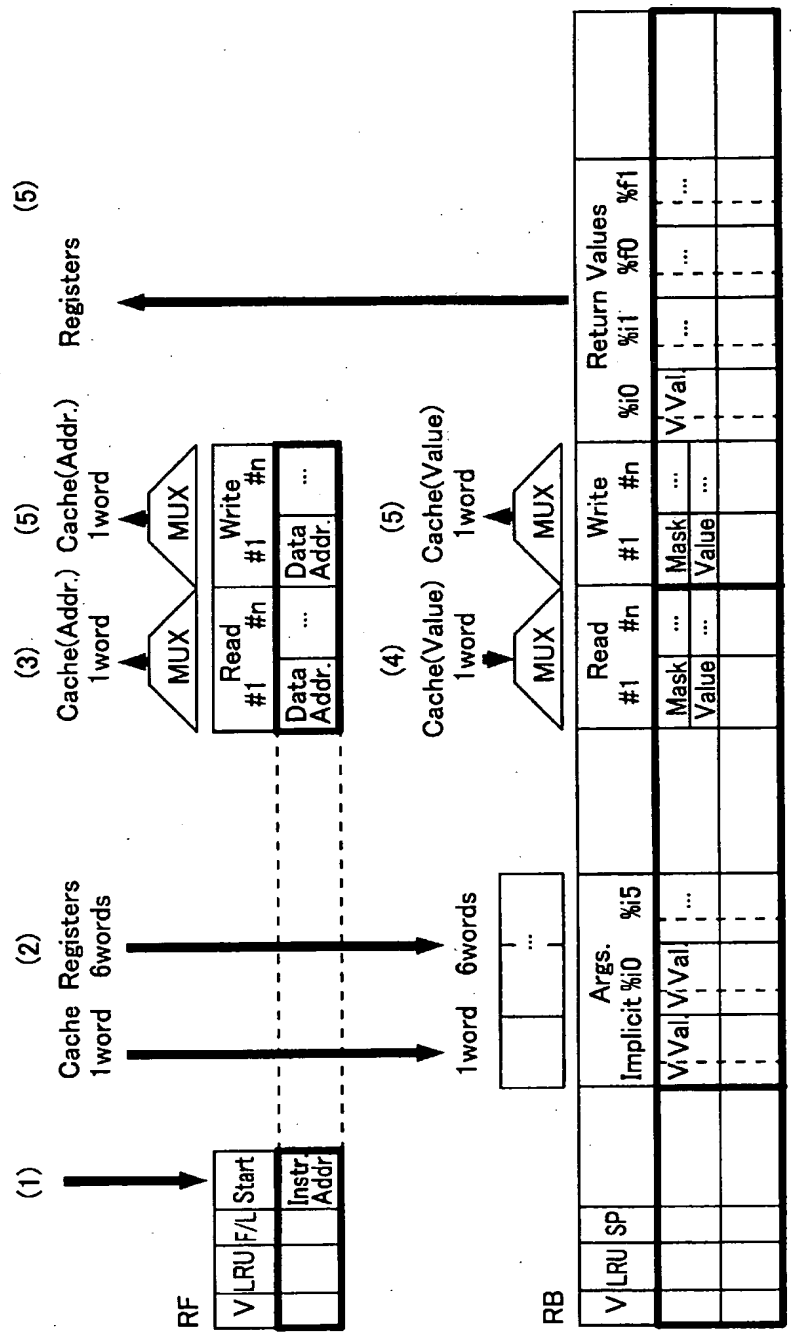
[図46(b)]


Data area
LIMIT
SP(in B)
SP(in A)
Stack area
Globals
Locals-B
Ret.Val.
Args
Locals-A
A/B/Cin/out
Cin/out
B/Cin/out

[図47]


Low-Addr.
Valid Data
LIMIT
Instructions
Global Vars.
%sp
%sp+64
%o0
%o5
%sp+92
Valid Data in Func.A
Implicit Arg.
Explicit Arg.#1
Explicit Arg.#6
Explicit Arg.#7
(a) Call A.
%fp
%fp+64
%i0
%i5
%fp+92
Local Vars.
Func. A
Args
(b) Call/Return B.
%fp+64
%i0
Local Vars.
Func. B
Retv
Args
(c) Return A.
High-Addr.

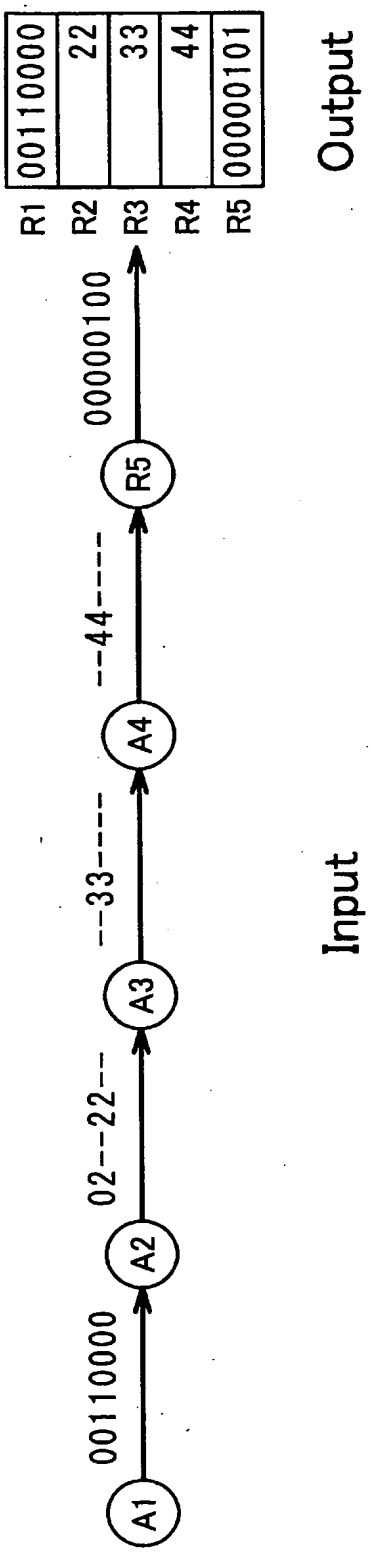
[図48]


(1)
(2)
Cache 1word
Registers 6words
(3)
Cache(Addr.) 1word
(4)
Cache(Value) 1word
(5)
Cache(Addr.) 1word
(5)
Cache(Value) 1word
(5)
Registers
MUX
MUX
MUX
MUX
RF
V
LRU
F/L
Start
Instr. Addr
Read #1 #n
Data Addr.
Write #1 #n
Data Addr.
RB
V
LRU
SP
Args. Implicit %i0 %i5
V
Val.
V
Val.
Read #1 #n
Mask
Value
Write #1 #n
Mask
Value
Return Values %i0 %i1 %f0 %f1
V
Val.
1word 6words

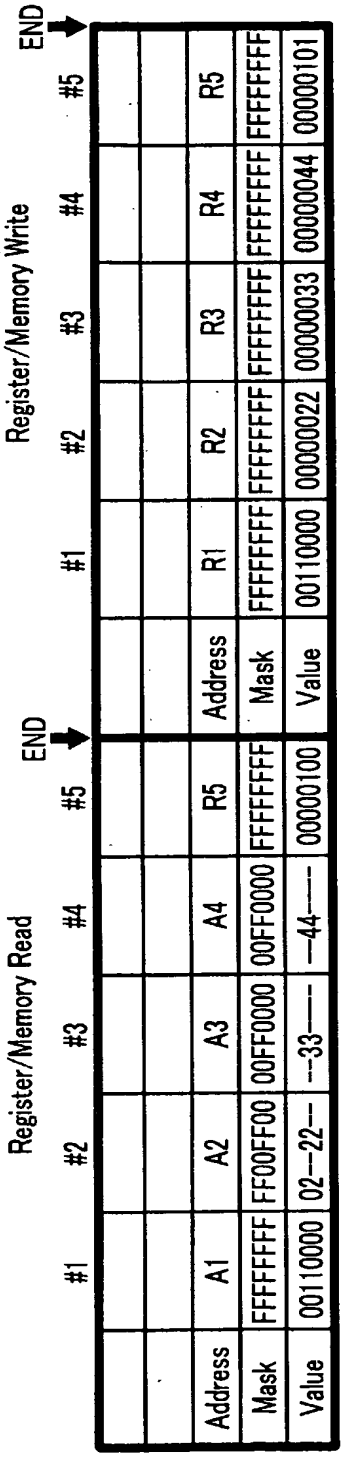
[図49]

```
PC:1000
 set A1->R0
 ld (R0)->R1
 set A2->R0
 ldb (R0)->R2
 ldb (A2+R2)->R2
 set A3->R0
 ldb (R0)->R3
 ldb (A4=R1+R2)->R4
 add R5+1->R5
```

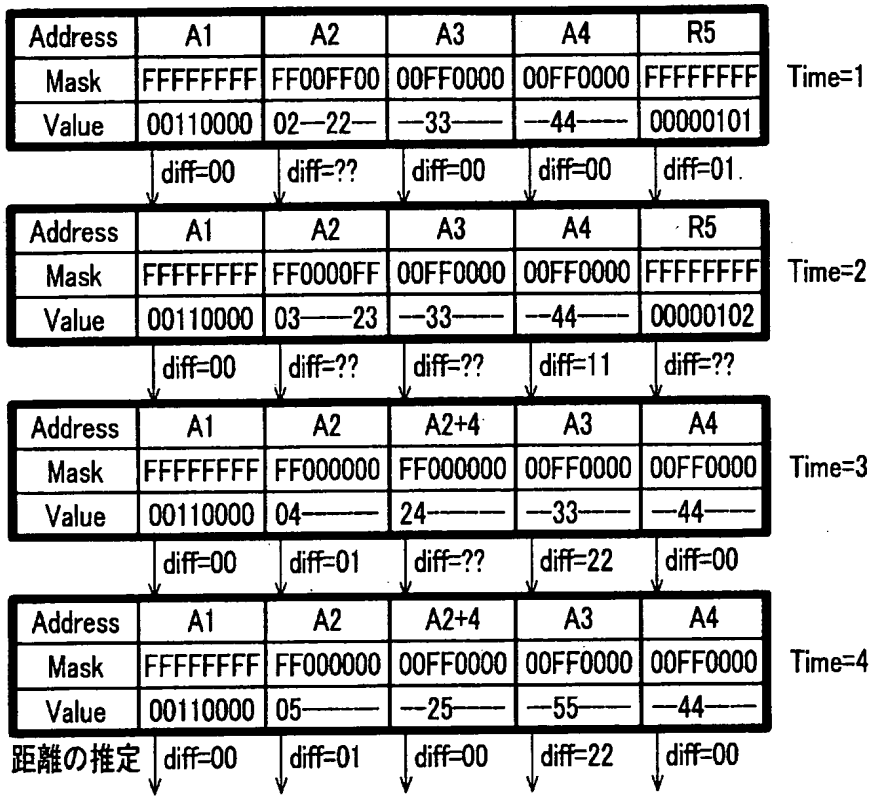
[図50]


R1
R2
R3
R4
R5
00110000
22
33
44
00000101
Output
00000100
00110000
02--22--
--33----
--44----
A1
A2
A3
A4
R5
Input

[図51]


Register/Memory Read
END
Register/Memory Write
END
#1
#2
#3
#4
#5
Address
A1
A2
A3
A4
R5
Mask
FFFFFFFF
FF00FF00
00FF0000
00FF0000
FFFFFFFF
Value
00110000
02--22--
--33----
--44----
00000100
Address
R1
R2
R3
R4
R5
Mask
FFFFFFFF
FFFFFFFF
FFFFFFFF
FFFFFFFF
FFFFFFFF
Value
00110000
00000022
00000033
00000044
00000101

[図52]



[図53]

Address	A1	A2	A2+4	A3	A4	
Mask	FFFFFFFF	FF000000	00FF0000	00FF0000	00FF0000	予測距離=1
Value	00110000	06———	—25——	—77——	—44——	

Address	A1	A2	A2+4	A3	A4	
Mask	FFFFFFFF	FF000000	00FF0000	00FF0000	00FF0000	予測距離=2
Value	00110000	07———	—25——	—99——	—44——	

Address	A1	A2	A2+4	A3	A4	
Mask	FFFFFFFF	FF000000	00FF0000	00FF0000	00FF0000	予測距離=3
Value	00110000	08———	—25——	—BB——	—44——	

Address	A1	A2	A2+4	A3	A4	
Mask	FFFFFFFF	FF000000	00FF0000	00FF0000	00FF0000	予測距離=4
Value	00110000	09———	—25——	—DD——	—44——	

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2005/005591

A. CLASSIFICATION OF SUBJECT MATTER
Int.Cl[7] G06F9/38, 9/40

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
Int.Cl[7] G06F9/30-9/42

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Jitsuyo Shinan Koho 1922-1996 Jitsuyo Shinan Toroku Koho 1996-2005
Kokai Jitsuyo Shinan Koho 1971-2005 Toroku Jitsuyo Shinan Koho 1994-2005

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	Yasuhiko NAKAJIMA et al., "Kansuchi Sairiyo Oyobi Heiretsu Jizen Jikko ni yoru Kosokuka Gijutsu", Transactions of Information Processing Society of Japan: High Performance Computing System, 15 September, 2002 (15.09.02), Vol.43, No.SIG6 (HPS5), pages 1 to 12	1-31
A	JP 2002-318688 A (NEC System Technology Kabushiki Kaisha), 31 October, 2002 (31.10.02), (Family: none)	1-31
A	WO 99/45463 A1 (Hitachi, Ltd.), 10 September, 1999 (10.09.99), & US 6810474 B1	1-31

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
14 April, 2005 (14.04.05)	10 May, 2005 (10.05.05)
Name and mailing address of the ISA/ Japanese Patent Office	Authorized officer
Facsimile No.	Telephone No.

Form PCT/ISA/210 (second sheet) (January 2004)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2005/005591

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JP 11-212788 A (Toshiba Corp.), 06 August, 1999 (06.08.99), & US 6415380 B1	1-31